**PROJECT REPORT**

**ON**

**Vehicle Renting System**

**Submitted for partial fulfillment for Award of**

**BACHELOR OF COMPUTER APPLICATION**

**By**

**Shantanu SINGH**

(**ROLLNO-17072010083**)

**UNDER THE GUIDANCE**

**OF**

**Mr. Sunil Kumar Singh**

**Head, Department of Computer Applications, SCPM, Lucknow**

****

**Sherwood College of Professional Management**

****

**DEPARTMENT OF COMPUTER SCIENCE**

**UNIVERSITY OF LUCKNOW**

**LUCKNOW -226007**

**AUGUST-2018**

Preface

**Preface-**

I am glad to make this project **Vehicle Renting System**and now I make this project report. In this project report, I include about my project, objective of project, lists of project activities etc.

                       Doing this project, I have a great experience and I get knowledge about boutique system. These all experience and knowledge I am going to share by my project report.

                    During work on this project I met many technical problems. These technical problems are solved by my Teacher **Sir Sunil Kumar Singh.** He is very helpful for me in this project so I am very thankful to him.

**Thanking you.**

Acknowledgement

**ACKNOWLEDGEMENT: -**

Fifth Semester Project is a major component of Academic Schedule of B.C.A. Hence, I worked on **Vehicle Renting System.** The conceptual Knowledge acquired by Management/ Computer student is best manifested in the project they undergo. The present project gives a perfect vent to my understanding of the practicalities of information of different Educational areas. I express my whole-hearted gratitude towards **Sherwood College of Professional Management** for having given me the opportunity to undergo my project in the field of Website development of great report and allowing me to gain invaluable experience. I express my heart-felt gratitude to **Mr. Sunil Kumar Singh**  for supervising me during the project period. I also express my special thanks to all the staff member who gave me their precious time and help me whenever required.

I am also grateful to my parents who have always been supportive in giving me correct decision and advice.

I also express my sincere thanks to all the Respondents, without whose help the completion of this project report was not possible

(Signature of Student)

|  |  |  |
| --- | --- | --- |
| **S.no** | **Title** | **Pg. No.** |
| 1. | Introduction | 8 |
| 1.1 | Background | 9 |
| 1.2 | Objective | 9-12 |
| 1.3 | Purpose & Scope | 12-13 |
| 1.3.1 | Purpose | 12 |
| 1.3.2 | Scope | 13 |
| 2. | Survey of technology | 14-17 |
| 3. | Requirement & Analysis | 18 |
| 3.1 | Problem Definition | 19 |
| 3.2 | Planning & Scheduling | 19-20 |
| 3.3 | Requirement Specification | 20 |
| 3.3.1 | Software | 20 |
| 3.3.2 | Hardware | 20 |
| 3.4 | Preliminary Product Description | 20-22 |
| 3.5 | Conceptual Modules | 22-34 |
| 4. | Project Code | 35-113 |
| 5. | Design Screenshots | 114-145 |
| 6. | Testing | 146-150 |
| 7. | Future & Scope | 151-153 |
| 8. | Bibliography | 154-156 |

**Structure**

REPORT

**(VEHICLE RENTING SYSTEM)**

Introduction

**1.1 Background: -**

Nowadays, there are online vehicle reservations which give much benefit to user. A rental service is a service in which customers arrive to request the hire of a rental unit. It is more convenient than carrying the cost of owning and maintaining the unit.

This project is designed so as to be used by Vehicle Rental Company specializing in renting vehicles to customers. It is an online system through which customers can view available vehicles, register, view profile and book vehicle.

A vehicle rental or vehicle hire agency is a company that rents automobiles for short period of time for a fee whether in a few hours or a few days or week. It is an extended form of a rental shop, often organized with numerous local branches (which allow a user to return a vehicle to a different location), and primarily located near airports or busy city areas and often complemented by a website allowing online reservations.

Vehicle rental agencies primarily serve people who have vehicles that are temporarily out of reach or out of service, for example travelers who are out of town or owners of damaged or destroyed vehicles who are awaiting repair or insurance compensation.

Because of the variety of sizes of the vehicles, our company serves the self-moving industry needs, by renting vans or trucks, and in certain markets other types of vehicles such as motorcycles or scooters are also offered. In short, it is a system design especially for large, premium and small vehicle rental business. The vehicle rental system provides complete functionality of listing and booking vehicles.

**1.2 Objective: -**

Today’s world is computer world because most of work is doing with the help of computer. Dependency on computer is behind the few reasons. We cannot easily manage to store large number of data or information single handle. If we will be needing some information or data in urgency then we cannot manage in manually these works are very difficult if we cannot use computer.

This software is basically updating the manual chemist Inventory system to automated Inventory system. So that organization can manage their record in efficient and organize them.

• The main objective is to automate non-computer environment

• To save manpower.

• It will speed the processing of data and transaction.

• It will provide best security features such as provisions of passwords

• To transform the manual process of hiring vehicle to a computerize system.

• To validate the Rental vehicle system using user satisfaction test.

• To produce the documentation such as Software Requirement Specification (SRS), Software Design Description (SDD) as system development reference.

• To produce a web-based system that allow customer to register and reserve vehicle online and for the company to effectively manage their vehicle rental business.

• To ease customer’s task whenever they need to rent a vehicle.

1. **System Objective: -**

Today’s world is computer world because most of work is doing with the help of computer. Dependency on computer is behind the few reasons. We cannot easily manage to store large number of data or information single handle.

If we will be needing some information or data in urgency then we cannot manage in manually these works are very difficult if we cannot use computer.

1. **System Context: -**

This section clearly depicts the environment and boundaries of the Vehicle Rental System and the entities with which it interacts. It helps us see how the system fits into the existing scheme of things. What the system will do by itself.

1. **Functional Requirement: -**

This Software must request Username and Password for access to data, only after authentication will allow access to the system. The Software must allow input of products data from administrator and secured access.

Requirement analysis is a software engineering technique that is composed of the various tasks that determine the needs or conditions that are to be met for a new or altered product, taking into consideration the possible conflicting requirements of the various users.

Functional requirements are those requirements that are used to illustrate the internal working nature of the system, the description of the system, and explanation of each subsystem. It consists of what task the system should perform, the processes involved, which data should the system holds and the interfaces with the user.

The functional requirements identified are:

a) **Customer’s registration:** The system should allow new users to register online and generate membership card.

b) **Online reservation of vehicles:** Customers should be able to use the system to make

c) Booking and online reservation.

d) **Automatic update to database once reservation is made or new customer registered:** Whenever there’s new reservation or new registration, the system should be able update the database without any additional efforts from the admin.

e) **Feedbacks to customers:** It should provide means for customers to leave feedback.

1. **Non-Functional Requirement: -**

In this Software Input error will be returned in red with appropriate message box. System should automatically update after every transaction. It describes aspects of the system that are concerned with how the system provides the functional requirements. They are:

a) **Security:** The subsystem should provide a high level of security and integrity of the data held by the system, only authorized personnel of the company can gain access To the Company’s secured page on the system; and only users with valid password and username can login to view user’s page.

b) **Performance and Response time:** The system should have high performance rate when executing user’s input and should be able to provide feedback or response within a short time span usually 50 seconds for highly complicated task and 20 to 25seconds for less complicated task.

c) **Error handling:** Error should be considerably minimized and an appropriate error message that guides the user to recover from an error should be provided. Validation of user’s input is highly essential. Also, the standard time taken to recover from an error should be 15 to 20 seconds.

d) **Availability:** This system should always be available for access at 24 hours, 7 days awake. Also, in the occurrence of any major system malfunctioning, the system should be available in 1 to 2 working days, so that the business process is not severely affected.

e) **Ease of use:** Considered the level of knowledge possessed by the users of this system, a simple but quality user interface should be developed to make it easy to understand and required less training.

**1.3 Purpose and Scope**

**1.3.1 Purpose: -**

The purpose of this document is to specify requirements and to give guidelines for the development of above said project. In particular it gives guidelines on how to prepare the above said project.

The advancement in Information Technology and internet penetration has greatly enhanced various business processes and communication between companies (services provider) and their customers of which car rental industry is not left out.

This Vehicle Rental System is developed to provide the following services:

• **Enhance Business Processes:** To be able to use internet technology to project the rental company to the global world instead of limiting their services to their local domain alone, thus increase their return on investment (ROI).

• **Online Vehicle Reservation:** Tools through which customers can reserve available vehicles online prior to their expected pick-up date or time.

• **Customer’s registration:** A registration portal to hold customer’s details, monitor their transaction and used same to offer better and improve services to them.

• **Group bookings:** Allows the customer to book space for a group in the case of weddings or corporate meetings (Event management).

**1.3.2 Scope: -**

This project traverses a lot of areas ranging from business concept to computing field and required to perform several researches to be able to achieve the project objectives.

The area covers include:

•**Car rental industry:** This includes study on how the car rental business is being done, process involved and opportunity that exist for improvement.

• Java Technology used for the development of the application.

• General customers as well as the company’s staff will be able to use the system effectively.

• Web-platform means that the system will be available for access 24/7 except when there is a temporary server issue which is expected to be minimal

• Eco-friendly: The monitoring of the vehicle activity and the overall business becomes easy and includes the least of paper work.

• The software acts as an office that is open 24/7. It increases the efficiency of the management at offering quality services to the customers.

• It provides custom features development and support with the software.

SURVEY OF TECHNOLOGIES

**Java: -**

Java is a general-purpose computer-programming language that is concurrent, class-based, object-oriented and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. As of 2016, Java is one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licenses. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java (bytecode compiler), GNU Class path (standard libraries), and Iced Tea-Web (browser plug-in for applets).

The latest version is Java 10, released on March 20, 2018, which follows Java 9 after only six months in line with the new release schedule. Java 8 is still supported but there will be no more security updates for Java 9. Versions earlier than Java 8 are supported by companies on a commercial basis; e.g. by Oracle back to Java 6 as of October 2017 (while they still "highly recommend that you uninstall" pre-Java 8 from at least Windows computers).

**Versions**

As of 20 March 2018, both Java 8 and 10 are officially supported. Major release versions of Java, along with their release dates:

* JDK 1.0 (January 23, 1996)
* JDK 1.1 (February 19, 1997)
* J2SE 1.2 (December 8, 1998)
* J2SE 1.3 (May 8, 2000)
* J2SE 1.4 (February 6, 2002)
* J2SE 5.0 (September 30, 2004)
* Java SE 6 (December 11, 2006)
* Java SE 7 (July 28, 2011)
* Java SE 8 (March 18, 2014)
* Java SE 9 (September 21, 2017)
* Java SE 10 (March 20, 2018)

**MySQL: -**

MySQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. For proprietary use, several paid editions are available, and offer additional functionality.

MySQL is a central component of the LAMP open-source web application software stack (and other "AMP" stacks). LAMP is an acronym for "Linux, Apache, MySQL, and Perl/PHP/Python". Applications that use the MySQL database include: TYPO3, MODx, Joomla, WordPress, Simple Machines Forum, phpBB, MyBB, and Drupal. MySQL is also used in many high-profile, large-scale websites, including Google (though not for searches), Facebook, Twitter, Flickr, and YouTube.

**Release history**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Release** | **General availability** | **Latest minor version** | **Latest release** | **End of Support** |
| **5.1** | November 14, 2008; 9 years ago | 5.1.73 | 2013-12-03 | December 2013 |
| **5.5** | December 3, 2010; 7 years ago | 5.5.60 | 2018-04-19 | December 2018 |
| **5.6** | February 5, 2013; 5 years ago | 5.6.40 | 2018-04-19 | February 2021 |
| **5.7** | October 21, 2015; 2 years ago | 5.7.22 | 2018-04-19 | October 2023 |
| **8.0** | April 19, 2018; 4 months ago | 8.0.11 | 2018-04-19 | N/A |

Requirements and Analysis

**3.1 Problem Definition: -**

In a vehicle rental service, a vehicle that can be used temporarily by paying a fee during a specified period. Getting a rental vehicle helps people get around despite the fact they do not have access to their own personal vehicle or don't own a vehicle at all.

The individual who needs a car must contact a rental vehicle company and contract out for a vehicle. This system increases customer retention and simplify vehicle and staff management.

**3.2 Planning and Scheduling: -**

**PERT CHART**

A project plan needs to be created to ensure the timely completion of the project. As part of project analysis, we break the project down to a number of stages and use a Gantt chart and PERT chart to describe specific tasks and status. The Work Breakdown Structure of our proposed system “Vehicle Rental System” is shown below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Task Name** | **Duration** | **Start** | **Finish** | **Predecessor** | **Resource Names** |
| 1 | Project Initiation | 3 days | 1/7/18 | 3/7/18 |  |  |
| 2 | Draft Project Plan | 4 days | 4/7/18 | 7/7/18 |  |  |
| 3 | Analysis Phase | 9 days | 8/7/18 | 16/7/18 |  |  |
| 4 | Plan User Interviews | 2 days | 17/7/18 | 18/7/18 |  |  |
| 5 | Schedule users Interviews | 4 days | 19/7/18 | 22/7/18 |  |  |
| 6 | Conducting users Interviews | 3 days | 23/7/18 | 25/7/18 |  |  |
| 7 | System Design | 15 days | 26/7/18 | 9/8/18 | 6 |  |
| 8 | Modules Design | 11 days | 10/8/18 | 20/8/18 |  |  |
| 9 | Data Structure Design | 4 days | 21/8/18 | 24/8/18 | 8 |  |
| 10 | User Interface Design | 4 days | 25/8/18 | 28/8/18 |  |  |
| 11 | Coding Phase | 36 days | 29/8/18 | 4/10/18 |  |  |
| 12 | Testing Phase | 9 days | 5/10/18 | 13/10/18 |  |  |
| 13 | Integration Testing | 6 days | 14/10/18 | 19/10/18 |  |  |
| 14 | System Level Testing | 5 days | 20/10/18 | 24/10/18 |  |  |
| 15 | Implementation | 4 days | 25/10/18 | 28/10/18 |  |  |
| 16 | Post-Implementation Review | 2 day | 29/10/18 | 30/10/18 |  |  |

**3.3 Requirements Specification: -**

**3.3.1 *Software Requirements***

1. Java/JDK
2. Net Beans
3. MySQL
4. SQL YOG

**3.3.2 *Hardware Requirements***

* Pentium IV Processor
* 512 MB RAM
* 40 GB HDD
* Color Monitor
* Keyboard, Mouse

**3.4 Preliminary Product Description: -**

It would be a multi user account system in key features. There will three types of main modules in the system

• Admin

• Customer

• Invoice Management

**AdMIN**

Like every other management system, the vehicle rental management system will have the admin. The admin will the entity that will monitor the activities and the records of whole system.

Following are some main facts related to the admin of the system. There will be only one admin in the system. Admin can view other user’s profile. The admin will have the power to delete any other users form the records or update the data of any other users. Any vehicle or the payment deal will be approved by the admin.

**CUSTOMERS**

Customers are the reason why I feel to introduce the vehicle rental management system, to make their journey wonderful, to get them fit for the environment they are traveling into.

**View the Vehicles –**

You, as a customer, can observe the lists of vehicles available in the inventory. The user can filter the records of the vehicle based on:

**Price -** The budget is an important factor. It will be easier to choose a vehicle rather than wondering what if I choose this vehicle and the price is higher. No tension at all, you can analyze the vehicle record and choose your best vehicle.

**Popular Vehicles:** If you want to take the vehicle which is popular in the system rather than thinking about the fact how this vehicle would perform, you better look into it. The already registered customers have given the feedback of their vehicle driving experience.

**Vehicle Brand:** If you are into brand, you can view the vehicles of your favorite brand. I have taken that too into the account. The vehicle brand can be BMW, Mercedes, Aston Martin, Honda, Mahindra etc. Just pick your pick.

**Rent a Vehicle:** After you have selected your favorite vehicle, you can fill the vehicle rental form which is available online. You just have to fill some details like for how many days you want to rent the vehicle, or if you want to rent on hourly basis, the vehicle details of the vehicle model you want to rent. After completing the rental form, you can pay suitable amount using net banking, your credit / debit vehicle.

**Return a rented Vehicle:** The customer can return a vehicle and if all the payments are cleared and the parts of the vehicles are not damaged, a number will be provided to the customer so that the customer can enter that number into the return vehicle section and the record is cleared from the rent a vehicle system and is moved to rental vehicle history.

**View Rental History:** You as a user can view the history of the vehicle you have rented in the vehicle rental management system. You can keep track of the amount you have spent, the vehicle you have driven, the number of vehicles you have rented etc.

**Feedback:** You as a user can share your experience with the vehicle rental management system. How much you loved it, or hated it. You can give the stars and provide some comments so that the dealer can assist the customers to their best capability they can.

**INVOICE MANAGEMENT**

After the amount is paid by the customers, the invoice department will generate the bill of the vehicle used and will reflect into the customers’ account. This department will also keep the receipts of newly vehicle is brought to the system so that it can further be used for analysis purpose.

**3.5 CONCEPTUAL MODELS SYSTEM**

**ARCHETECTURE DESIGN**

**1 DEFINING A SYSTEM: -**

Collection of components, which are interconnected, and work together to realize some objective, from a system. There are three components in every system, namely input, processing and output

Processing

Input Output

**SYSTEM DEVELOPMENT LIFE CYCLE**

The System development life cycle (SDLC), or Software development processing systems engineering, information systems and software engineering, is a process of creating or altering information systems, and the models and methodologies that people use to develop these systems. In software engineering, the SDLC concept underpins many kinds of software development methodologies.

These methodologies form the framework for planning and controlling the creation of an information system the process. Broadly, following are the different activities to be considered while defining the system development life cycle for the said project:

* Problem Definition
* System Analysis
* Study of existing system
* Drawback of the existing system
* Proposed system
* System Requirement study
* Data flow analysis
* Feasibility study
* System design
* Input Design (Database & Forms)
* Updating
* Query /Report design
* Administration
* Testing
* Implementation
* Maintenance

**2 SYSTEM ANALYSIS: -**

Systems analysis is the study of sets of interacting entities, including computer systems analysis. This field is closely related to requirements analysis or operations research. It is also "an explicit formal inquiry carried out to help someone (referred to as the decision maker) identify a better course of action and make a better decision than he might otherwise have made.

System development can generally be thought of having two major components: systems analysis and systems design. In System Analysis more emphasis is given to understanding the details of an existing system or a proposed one and then deciding whether the proposed system is desirable or not and whether the existing system needs improvements.

Thus, system analysis is the process of investigating a system, identifying problems, and using the information to recommend improvement to the system.

**3 SYSTEM DESIGN: -**

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering. If the broader topic of product development "blends the perspective of marketing, design, and manufacturing into a single approach to product development," then design is the act of taking the marketing information and creating the design of the product to be manufactured. Systems design is therefore the process of defining and developing systems to satisfy specified requirements of the user. Until the 1990s systems design had a crucial and respected role in the data processing industry. In the 1990s standardization of hardware and software resulted in the ability to build modular systems. The increasing importance of software running on generic platforms has enhanced the discipline of software engineering.

Object-oriented analysis and design methods are becoming the most widely used methods for computer systems design. The UML has become the standard language in object-oriented analysis and design. It is widely used for modeling software systems and is increasingly used for high designing non-software systems and organizations.

**ENTITY RELATION DIAGRAM**

The Entity Relation Model or Entity Relation Diagram (ERD) is a data model or diagram for high-level description of conceptual data model, and it provides a graphical notation for representing such data models in the form of entity relationship diagrams. Such models are typically used in the first stage of Management information system design; they are used for example, to describe information needs and/ or the type of information that is to be stored in the Database during the requirement analysis.

The data modeling technique, however, can be used to describe any ontology (i.e. an overview and classification of used term and their relationships) for a certain universe of discourse (i.e. area of interest).

In the case of design, a Management Information System that is based on a database, the conceptual data model is, a later stage (usually called logical design), mapped to a logical data model such as, relational data model; this is turn in mapped to a physical model during physical design.

Note that sometimes, both of the phases are referred a “physical design”. There are number of conventions for entity-relation diagrams (ERDs). The classical notation is described in the remainder of this article, and mainly related to the conceptual modeling. There is a range of notation more typically employed in physical and logical database design.

MANAGES

VEHICLE DETAIL

ADMIN

VEHICLE REPORT

CUSTOMER

MANAGES

HAS

RENT WITH

VEHICLE RENTAL

**DATA FLOW DIAGRAM**

The data flow diagram shows the flow of data within any system. It is an important tool for designing phase of software engineering. Larry Constantine first developed it. It represents graphical view of flow of data. It’s also known as BUBBLE CHART. The purpose of DFD is major transformation that will become in system design symbols used in DFD.

In the DFD, four symbols are used and they are as follows.

1. A square defines a source (originator) or destination of system data.
2. An arrow identifies data flow-data in motion. It is 2a pipeline through which information flows.
3. A circle or a “bubble “(Some people use an oval bubble) represents a process that transfers informing data flows into outgoing data flows.
4. An open rectangle is a data store-data at rest, or a temporary

Repository of data.

**Context Level Data Flow Diagram: -**

This level shows the overall context of the system and its operating environment and shows the whole system as just one process. Online book store is shown as one process in the context diagram; which is also known as zero level DFD, shown below.

VEHICLE

CUSTOMER

REQUEST

ADMIN

**Zero Level Data Flow Diagram**

The context diagram plays important role in understanding the system and determining the boundaries. The main process can be broken into sub-processes and system can be studied with more detail; this is where 1st level DFD comes into play.

**First Level DFD: -**

This level (level 1) shows all processes at the first level of numbering, data stores, external entities and the data flows between them. The purpose of this level is to show the major high-level processes of the system and their interrelation. A process model will have one, and only one, level-1 diagram. A level-1 diagram must be balanced with its parent context level diagram, i.e. there must be the same external entities and the same data flows, these can be broken down to more detail in the level 1.

PAY BILL

**CUSTOMER**

BUY OR ASK

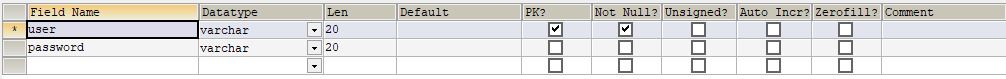
Rental system

ADMIN

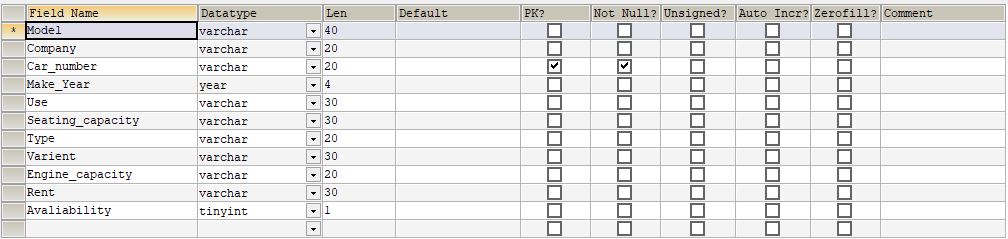
**One level data flow diagram**

**Data Tables**

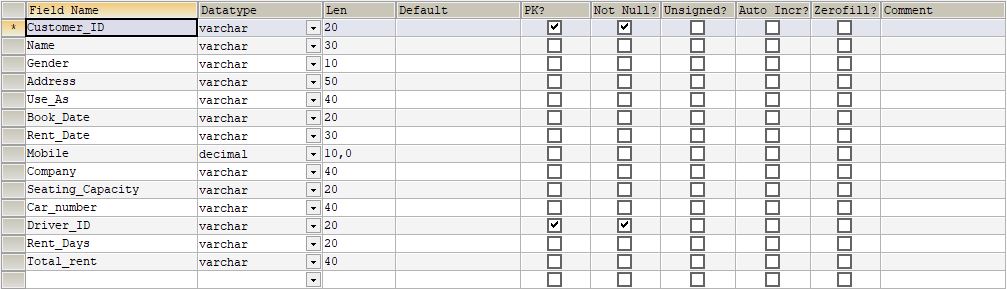
**//Login Table**



**//ADD VEHICLE Table**



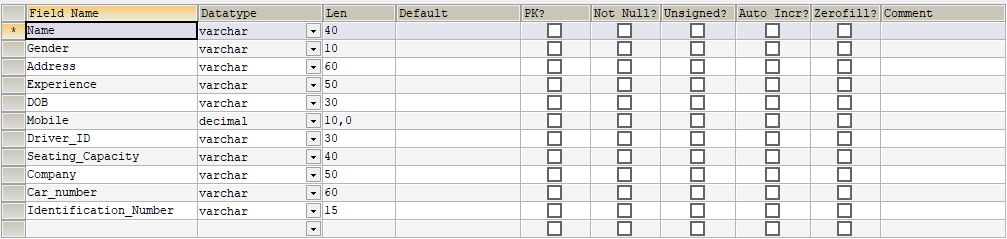
**//ADD CUSTOMER Table**



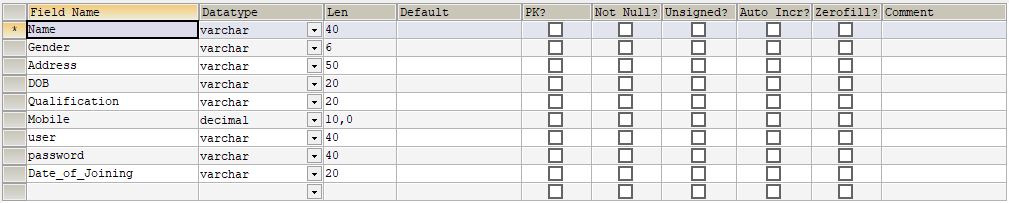
**//ADD COMPANY Table**



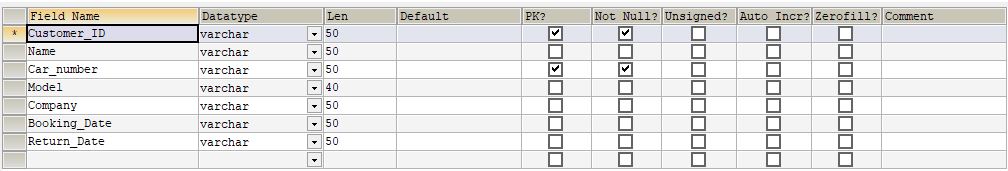
**//ADD DRIVER Table**



**//ADD EMPLOYEE TABLE**



**//RETURN CAR TABLE**



**Login form**

**Login: -**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name Column** | **Data Type** | **Size** | **Constraints** |
| User | Varchar | 20 | Primary key |
| Password | Varchar | 20 | Not Null |

**add\_company: -**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name Column** | **Data Type** | **Size** | **Constraints** |
| Company\_name | Varchar | 40 | Primary key |

**add\_customer: -**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name Column** | **Data Type** | **Size** | **Constraints** |
| Customer\_ID | Varchar | 20 | Primary key |
| Name | Varchar | 30 | Not Null |
| Gender | Varchar | 10 | Not Null |
| Address | Varchar | 50 | Not Null |
| Use\_As | Varchar | 40 | Not Null |
| Book\_Date | Varchar | 20 | Not Null |
| Rent\_Date | Varchar | 30 | Not Null |
| Mobile | decimal | 10,0 | Not Null |
| Company | Varchar | 40 | Not Null |
| Seating\_Capacity | Varchar | 20 | Not Null |
| Car\_number | Varchar | 40 | Not Null |
| Driver\_ID | Varchar | 20 | Primary key |
| Rent\_Days | Varchar | 20 | Not Null |
| Total\_rent | Varchar | 40 | Not Null |

**add\_driver: -**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name Column** | **Data Type** | **Size** | **Constraints** |
| Name | Varchar | 40 | Not Null |
| Gender | Varchar | 10 | Not Null |
| Address | Varchar | 60 | Not Null |
| Experience | Varchar | 50 | Not Null |
| DOB | Varchar | 30 | Not Null |
| Mobile | Decimal | 10,0 | Not Null |
| Driver\_ID | Varchar | 30 | Not Null |
| Seating\_Capacity | Varchar | 40 | Not Null |
| Company | Varchar | 50 | Not Null |
| Car\_number | Varchar | 60 | Not Null |
| Identification\_Number | Varchar | 15 | Not Null |

**add\_emp: -**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name Column** | **Data Type** | **Size** | **Constraints** |
| Name | Varchar | 40 | Not Null |
| Gender | Varchar | 6 | Not Null |
| Address | Varchar | 50 | Not Null |
| DOB | Varchar | 20 | Not Null |
| Qualification | Varchar | 20 | Not Null |
| Mobile | Decimal | 10,0 | Not Null |
| User | Varchar | 40 | Not Null |
| Password | Varchar | 40 | Not Null |
| Date\_of\_Joining | Varchar | 20 | Not Null |

**add\_vehicle: -**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name Column** | **Data Type** | **Size** | **Constraints** |
| Model | Varchar | 40 | Not Null |
| Company | Varchar | 20 | Not Null |
| Car\_number | Varchar | 20 | Primary key |
| Make\_Year | Year | 4 | Not Null |
| Use | Varchar | 30 | Not Null |
| Seating\_capacity | Varchar | 30 | Not Null |
| Type | Varchar | 20 | Not Null |
| Varient | Varchar | 30 | Not Null |
| Engine\_capacity | Varchar | 20 | Not Null |
| Rent | Varchar | 30 | Not Null |
| Avaliability | tinyint | 1 | Not Null |

CODING

**Database Connectivity: -**

**Coding: -**

package vehicle\_renting\_system;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.Statement;

public class DB {

public Connection con;

public PreparedStatement pstmt;

public Statement stmt;

public ResultSet rst;

public DB()

{

try

{

Class.forName("com.mysql.jdbc.Driver");

con=DriverManager.getConnection("jdbc:mysql://localhost:3306/renting","root","singh@323");

}

catch(Exception e)

{

e.printStackTrace();

}

}

}

**ANIMATION: -**

**CODING: -**

package vehicle\_renting\_system;

public class Vehicle\_Renting\_System

{

public static void main(String[] args)

{

splash sp=new splash();

sp.setVisible(true);

login l=new login();

try

{

for(int z=0;z<=100;z++)

{

Thread.sleep(30);

sp.jProgressBar1.setValue(z);

if(z==100)

{

sp.setVisible(false);

l.setVisible(true);

}

}

}

catch(Exception e)

{

e.printStactTrace();

}

}

}

**LOGIN: -**

**Coding On Login as Employee Button: -**

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt)

{

Driver.setVisible(true);

}

**Coding On Admin Button Coding: -**

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)

{

Employee.setVisible(true);

}

**Coding On Admin Login Button: -**

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)

{

try

{

String user;

user = t1.getText();

String password;

password = p1.getText();

DB dbc=new DB();

dbc.pstmt=dbc.con.prepareStatement("select \* from admin\_login where user=? And password=?");

dbc.pstmt.setString(1,user);

dbc.pstmt.setString(2,password);

dbc.rst=dbc.pstmt.executeQuery();

if(dbc.rst.next())

{

new login().setVisible(false);

Driver.setVisible(false);

new Driver().setVisible(true);

}

else

{

JOptionPane.showMessageDialog(null,"Incorrect Username Or Password");

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Coding On Admin Login Button: -**

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt)

{

try

{

String user;

user = t2.getText();

String password;

password = p2.getText();

DB dbc=new DB();

dbc.pstmt=dbc.con.prepareStatement("select \* from admin\_login where user=? and

password=?");

dbc.pstmt.setString(1,user);

dbc.pstmt.setString(2,password);

dbc.rst=dbc.pstmt.executeQuery();

if(dbc.rst.next())

{

new login().setVisible(false);

Employee.setVisible(false);

new renting\_system().setVisible(true);

JOptionPane.showMessageDialog(null,"Matched");

}

else

{

JOptionPane.showMessageDialog(null,"Incorrect Username Or Password");

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Employee Main Window Coding: -**

**ADD DRIVER: -**

**Event Performed On Menu Item:-**

Action Event performed(actionPerformed)

**Coding: -**

private void jMenuItem8ActionPerformed(java.awt.event.ActionEvent evt)

{

Add\_Driver.setVisible(true);

try

{

DB db=new DB();

db.pstmt=db.con.prepareStatement("select Company\_name from add\_company");

db.rst=db.pstmt.executeQuery();

while(db.rst.next())

{

jComboBox14.addItem(db.rst.getString(1));

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Car Available Button Coding: -**

private void jButton10ActionPerformed(java.awt.event.ActionEvent evt)

{

try

{

String number=jTextField23.getText();

String company=(String)jComboBox14.getSelectedItem();

String seater=(String)jComboBox15.getSelectedItem();

Vector<Vector<String>>data=new Vector<Vector<String>>();

data.removeAllElements();

Vector<String>header=new Vector<String>();

header.removeAllElements();

header.add("Model");

header.add("Make Year");

header.add("Use As");

header.add("Type");

header.add("Varient");

header.add("Engine Capacity");

header.add("Rent");

DB db=new DB();

db.stmt=db.con.createStatement();

db.rst=db.stmt.executeQuery("select \* from add\_vehicle where Car\_number='"+number+"' ");

while(db.rst.next())

{

Vector<String>data1=new Vector<String>();

data1.add(db.rst.getString(1));

data1.add(db.rst.getString(4));

data1.add(db.rst.getString(5));

data1.add(db.rst.getString(7));

data1.add(db.rst.getString(8));

data1.add(db.rst.getString(9));

data1.add(db.rst.getString(10));

data.add(data1);

jTable4.setModel(new javax.swing.table.DefaultTableModel(data,header));

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Coding On Submit Button: -**

private void jButton11ActionPerformed(java.awt.event.ActionEvent evt)

{

try

{

String name=jTextField20.getText();

String address=jTextArea3.getText();

String Experience=(String) jComboBox12.getSelectedItem();

String DOB=jTextField21.getText();

String Mobile=jTextField48.getText();

String number=jTextField22.getText();

if(jRadioButton5.isSelected())

{

gender="Male";

}

if(jRadioButton6.isSelected())

{

gender="Female";

}

String Driver\_ID=jTextField19.getText();

String Seating=(String) jComboBox15.getSelectedItem();

String company=(String)jComboBox14.getSelectedItem();

String car\_number=jTextField23.getText();

String message="";

int a=1;

if(name.equals(""))

{

message=message+"Name Can not be Empty ";

a=0;

}

if(address.equals(""))

{

message=message+" Address Field can not be Empty ";

a=0;

}

if(DOB.equals(""))

{

message=message+" DOB Field can not be Empty ";

a=0;

}

if(Mobile.equals(""))

{

message=message+" Mobile Field can not be Empty ";

a=0;

}

if(Driver\_ID.equals(""))

{

message=message+" Driver ID can not be Empty ";

a=0;

}

if(car\_number.equals(""))

{

message=message+" Car Number can not be Empty ";

a=0;

}

if(a==0)

{

JOptionPane.showMessageDialog(this, message);

}

if(a==1)

{

DB db=new DB();

db.pstmt=db.con.prepareStatement("insert into add\_Driver values(?,?,?,?,?,?,?,?,?,?,?)");

db.pstmt.setString(1, name);

db.pstmt.setString(2,gender);

db.pstmt.setString(3, address);

db.pstmt.setString(4, Experience);

db.pstmt.setString(5, DOB);

db.pstmt.setString(6, Mobile);

db.pstmt.setString(7, Driver\_ID);

db.pstmt.setString(8, Seating);

db.pstmt.setString(9, company);

db.pstmt.setString(10, car\_number);

db.pstmt.setString(11, number);

int i=db.pstmt.executeUpdate();

if(i>0)

{

JOptionPane.showMessageDialog(this, "Driver Added Successfully");

jTextField20.setText(null);

jTextArea3.setText(null);

jTextField21.setText(null);

jComboBox12.setSelectedItem("Select");

jTextField48.setText(null);

jTextField22.setText(null);

jTextField19.setText(null);

jComboBox15.setSelectedItem("Select");

jComboBox14.setSelectedItem("Select");

jTextField23.setText(null);

}

else

{

JOptionPane.showMessageDialog(this, "ERROR!!!!!!!!!!");

}

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Coding On Cancel Button: -**

private void jButton14ActionPerformed(java.awt.event.ActionEvent evt)

{

Add\_Driver.setVisible(false);

}

**Modify Driver: -**

**Event Performed On Menu Item:-**

Action Event performed(actionPerformed)

**Coding: -**

private void jMenuItem9ActionPerformed(java.awt.event.ActionEvent evt)

{

Modify\_Driver.setVisible(true);

}

**Coding On Search Button: -**

private void jButton15ActionPerformed(java.awt.event.ActionEvent evt)

{

String user2=jTextField24.getText();

try {

DB dbc=new DB();

dbc.pstmt=dbc.con.prepareStatement("select \* from add\_Driver where Driver\_ID='"+user2+"'");

dbc.rst=dbc.pstmt.executeQuery();

if(dbc.rst.next())

{

jTextField25.setText(dbc.rst.getString(1));

String s=dbc.rst.getString(2);

if(s.equals("Male"))

{

jRadioButton7.setSelected(true);

jRadioButton8.setSelected(false);

}

if(s.equals("Female"))

{

jRadioButton8.setSelected(true);

jRadioButton7.setSelected(false);

}

jTextArea4.setText(dbc.rst.getString(3));

String a=dbc.rst.getString(4);

for(int h=0;h<jComboBox16.getItemCount();h++)

{

if(a.equals(jComboBox16.getItemAt(h)))

{

jComboBox16.setSelectedIndex(h);

}

}

jTextField26.setText(dbc.rst.getString(5));

String q=dbc.rst.getString(6);

for(int h=0;h<jComboBox17.getItemCount();h++)

{

if(q.equals(jComboBox17.getItemAt(h)))

{

jComboBox17.setSelectedIndex(h);

}

}

jTextField27.setText(dbc.rst.getString(7));

}

} catch (Exception e) {

e.printStackTrace();

}

}

**Coding On Update Button: -**

private void jButton16ActionPerformed(java.awt.event.ActionEvent evt) {

String user2=jTextField24.getText();

String name=jTextField25.getText();

String address=jTextArea4.getText();

String DOB=jTextField26.getText();

String qualification=(String)jComboBox17.getSelectedItem();

String Mobile=jTextField27.getText();

String Experience=(String)jComboBox16.getSelectedItem();

if(jRadioButton7.isSelected())

{

gender="Male";

}

if(jRadioButton8.isSelected())

{

gender="Female";

}

try

{

DB db=new DB();

db.pstmt=db.con.prepareStatement("update add\_Driver set Name=?,Gender=?,Address=?,Experience=?,DOB=?,Qualification=?,Mobile=? where Driver\_ID='"+user2+"' ");

db.pstmt.setString(1, name);

db.pstmt.setString(2, gender);

db.pstmt.setString(3, address);

db.pstmt.setString(4,Experience);

db.pstmt.setString(5,DOB);

db.pstmt.setString(6, qualification);

db.pstmt.setString(7, Mobile);

int i=db.pstmt.executeUpdate();

if(i>0)

{

JOptionPane.showMessageDialog(this, "Driver Record Updated Successfully");

jTextField24.setText(null);

jTextField25.setText(null);

jTextArea4.setText(null);

jComboBox17.setSelectedItem("Select");

jTextField26.setText(null);

jTextField27.setText(null);

jComboBox16.setSelectedItem("Select");

jRadioButton7.setSelected(false);

jRadioButton8.setSelected(false);

}

else

{

JOptionPane.showMessageDialog(this, "ERROR!!!!!!!!!!");

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Coding On Delete Button: -**

private void jButton17ActionPerformed(java.awt.event.ActionEvent evt) {

try

{

String user2=jTextField24.getText();

DB db=new DB();

db.pstmt=db.con.prepareStatement("delete from add\_Driver where Driver\_ID='"+user2+"' ");

int i=db.pstmt.executeUpdate();

if(i>0)

{

JOptionPane.showMessageDialog(this, "Driver Record Deleted Successfully");

jTextField24.setText(null);

jTextField25.setText(null);

jTextArea4.setText(null);

jComboBox17.setSelectedItem("Select");

jTextField26.setText(null);

jTextField27.setText(null);

jComboBox16.setSelectedItem("Select");

jRadioButton7.setSelected(false);

jRadioButton8.setSelected(false);

}

else

{

JOptionPane.showMessageDialog(this, "Error!!!");

}

}

catch (Exception e)

{

e.printStackTrace();

}

}

**Driver List: -**

**Event Performed On Menu Item:-**

Action Event performed(actionPerformed)

**Coding: -**

Drivers\_List.setVisible(true);

try

{

Vector<Vector<String>>data=new Vector<Vector<String>>();

data.removeAllElements();

Vector<String>header=new Vector<String>();

header.removeAllElements();

header.add("Name");

header.add("Gender");

header.add("Address");

header.add("Experience");

header.add("DOB");

header.add("Mobile");

header.add("Driver ID");

header.add("Seating Capacity");

header.add("Company");

header.add("Car Number");

header.add("Driver License");

DB db=new DB();

db.stmt=db.con.createStatement();

db.rst=db.stmt.executeQuery("select \* from add\_Driver");

while(db.rst.next())

{

Vector<String>data1=new Vector<String>();

data1.add(db.rst.getString(1));

data1.add(db.rst.getString(2));

data1.add(db.rst.getString(3));

data1.add(db.rst.getString(4));

data1.add(db.rst.getString(5));

data1.add(db.rst.getString(6));

data1.add(db.rst.getString(7));

data1.add(db.rst.getString(8));

data1.add(db.rst.getString(9));

data1.add(db.rst.getString(10));

data1.add(db.rst.getString(11));

data.add(data1);

jTable2.setModel(new javax.swing.table.DefaultTableModel(data,header));

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**List Of Customer: -**

**Event Performed On Menu Item:-**

Action Event performed(actionPerformed)

**Coding: -**

private void jMenuItem12ActionPerformed(java.awt.event.ActionEvent evt)

{

Customer\_List.setVisible(true);

try

{

Vector<Vector<String>>data=new Vector<Vector<String>>();

data.removeAllElements();

Vector<String>header=new Vector<String>();

header.removeAllElements();

header.add("Customer ID");

header.add("Name");

header.add("Gender");

header.add("Address");

header.add("Use As");

header.add("Book Date");

header.add("Renting Date");

header.add("Mobile");

header.add("Company");

header.add("Seating Capacity");

header.add("Car Number");

header.add("Driver ID");

header.add("Renting Days");

header.add("Total Renting Amount");

DB db=new DB();

db.stmt=db.con.createStatement();

db.rst=db.stmt.executeQuery("select \* from add\_customer");

while(db.rst.next())

{

Vector<String>data1=new Vector<String>();

data1.add(db.rst.getString(1));

data1.add(db.rst.getString(2));

data1.add(db.rst.getString(3));

data1.add(db.rst.getString(4));

data1.add(db.rst.getString(5));

data1.add(db.rst.getString(6));

data1.add(db.rst.getString(7));

data1.add(db.rst.getString(8));

data1.add(db.rst.getString(9));

data1.add(db.rst.getString(10));

data1.add(db.rst.getString(11));

data1.add(db.rst.getString(12));

data1.add(db.rst.getString(13));

data1.add(db.rst.getString(14));

data.add(data1);

jTable6.setModel(new javax.swing.table.DefaultTableModel(data,header));

}

}

catch(Exception e)

{ e.printStackTrace();}}

**Add Customer: -**

**Event Performed On Menu Item:-**

Action Event performed(actionPerformed)

**Coding: -**

private void jMenuItem10ActionPerformed(java.awt.event.ActionEvent evt)

{

Add\_Customer.setVisible(true);

try

{

DB db=new DB();

db.pstmt=db.con.prepareStatement("select Company\_name from add\_company");

db.rst=db.pstmt.executeQuery();

while(db.rst.next())

{

jComboBox19.addItem(db.rst.getString(1));

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Coding On Car Available Button:-**

private void jButton18ActionPerformed(java.awt.event.ActionEvent evt) {

try

{

String number=jTextField30.getText();

String company=(String)jComboBox19.getSelectedItem();

String seater=(String)jComboBox20.getSelectedItem();

Vector<Vector<String>>data=new Vector<Vector<String>>();

data.removeAllElements();

Vector<String>header=new Vector<String>();

header.removeAllElements();

header.add("Model");

header.add("Make Year");

header.add("Use As");

header.add("Type");

header.add("Varient");

header.add("Engine Capacity");

header.add("Rent");

DB db=new DB();

db.stmt=db.con.createStatement();

db.rst=db.stmt.executeQuery("select \* from add\_vehicle where Car\_number='"+number+"' ");

while(db.rst.next())

{

Vector<String>data1=new Vector<String>();

data1.add(db.rst.getString(1));

data1.add(db.rst.getString(4));

data1.add(db.rst.getString(5));

data1.add(db.rst.getString(7));

data1.add(db.rst.getString(8));

data1.add(db.rst.getString(9));

data1.add(db.rst.getString(10));

data.add(data1);

jTable5.setModel(new javax.swing.table.DefaultTableModel(data,header));

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Coding On Calculate Button: -** private void jButton19ActionPerformed(java.awt.event.ActionEvent evt)

{

String days=jTextField32.getText();

String car\_number=jTextField30.getText();

int total;

try{

String rent = null;

int temp;

DB dbc=new DB();

dbc.pstmt=dbc.con.prepareStatement("select rent from add\_vehicle where Car\_number='"+car\_number+"'");

dbc.rst=dbc.pstmt.executeQuery();

if(dbc.rst.next())

{

rent=dbc.rst.getString(1);

}

temp=Integer.parseInt(rent);

total=Integer.parseInt(days)\*temp;

jTextField33.setText(Integer.toString(total));

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Coding On Submit Button: -**

private void jButton20ActionPerformed(java.awt.event.ActionEvent evt) {

String customer\_ID=jTextField28.getText();

String name=jTextField29.getText();

String address=jTextArea5.getText();

String use\_as=(String)jComboBox18.getSelectedItem();

String book=jTextField34.getText();

String Rent\_date=jTextField35.getText();

String mobile=jTextField36.getText();

String company=(String)jComboBox19.getSelectedItem();

String capacity=(String)jComboBox20.getSelectedItem();

String car\_number=jTextField30.getText();

String Driver\_ID=jTextField31.getText();

String Rent\_Days=jTextField32.getText();

String Total\_rent=jTextField33.getText();

if(jRadioButton9.isSelected())

{

gender="Male";

}

if(jRadioButton10.isSelected())

{

gender="Female";

}

try

{

DB db=new DB();

db.pstmt=db.con.prepareStatement("insert into add\_customer values(?,?,?,?,?,?,?,?,?,?,?,?,?,?)");

db.pstmt.setString(1, customer\_ID);

db.pstmt.setString(2, name);

db.pstmt.setString(3,gender);

db.pstmt.setString(4, address);

db.pstmt.setString(5, use\_as);

db.pstmt.setString(6, book);

db.pstmt.setString(7,Rent\_date);

db.pstmt.setString(8, mobile);

db.pstmt.setString(9, company);

db.pstmt.setString(10, capacity);

db.pstmt.setString(11, car\_number);

db.pstmt.setString(12, Driver\_ID);

db.pstmt.setString(13, Rent\_Days);

db.pstmt.setString(14, Total\_rent);

int i=db.pstmt.executeUpdate();

if(i>0)

{

JOptionPane.showMessageDialog(this, "Customer Added Successfully");

jTextField28.setText(null);

jTextField29.setText(null);

jTextArea5.setText(null);

jComboBox18.setSelectedItem("Select");

jComboBox19.setSelectedItem("Select");

jTextField34.setText(null);

jTextField35.setText(null);

jComboBox20.setSelectedItem("Select");

jTextField36.setText(null);

jTextField30.setText(null);

jTextField32.setText(null);

jTextField33.setText(null);

jTextField31.setText(null);

}

else

{

JOptionPane.showMessageDialog(this, "ERROR!!!!!!!!!!");

}

db.pstmt=db.con.prepareStatement("Update Avaliability from add\_vehicle where Car\_number='"+car\_number+"' ");

db.pstmt.setString(1,"0");

db.pstmt.executeUpdate();

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Coding On Cancel Button: -**

private void jButton21ActionPerformed(java.awt.event.ActionEvent evt)

{

Add\_Customer.setVisible(false);

}

**Modify Customer: -**

**Event Performed On Menu Item:-**

Action Event performed(actionPerformed)

**Coding: -**

private void jMenuItem11ActionPerformed(java.awt.event.ActionEvent evt)

{

Modify\_Customer.setVisible(true);

}

**Coding On Search By ID Button: -**

private void jButton25ActionPerformed(java.awt.event.ActionEvent evt)

{

String user2=jTextField40.getText();

try

{

DB dbc=new DB();

dbc.pstmt=dbc.con.prepareStatement("select \* from add\_customer where Customer\_ID='"+user2+"'");

dbc.rst=dbc.pstmt.executeQuery();

if(dbc.rst.next())

{

jTextField39.setText(dbc.rst.getString(2));

String s=dbc.rst.getString(3);

if(s.equals("Male"))

{

jRadioButton11.setSelected(true);

jRadioButton12.setSelected(false);

}

if(s.equals("Female"))

{

jRadioButton12.setSelected(true);

jRadioButton11.setSelected(false);

}

jTextArea6.setText(dbc.rst.getString(4));

jTextField37.setText(dbc.rst.getString(8));

}

}

catch (Exception e)

{

e.printStackTrace();

}

}

**Coding On Search By Name: -**

private void jButton22ActionPerformed(java.awt.event.ActionEvent evt) {

String user2=jTextField39.getText();

try {

DB dbc=new DB();

dbc.pstmt=dbc.con.prepareStatement("select \* from add\_customer where Name='"+user2+"'");

dbc.rst=dbc.pstmt.executeQuery();

if(dbc.rst.next())

{

jTextField40.setText(dbc.rst.getString(1));

String s=dbc.rst.getString(3);

if(s.equals("Male"))

{

jRadioButton11.setSelected(true);

jRadioButton12.setSelected(false);

}

if(s.equals("Female"))

{

jRadioButton12.setSelected(true);

jRadioButton11.setSelected(false);

}

jTextArea6.setText(dbc.rst.getString(4));

jTextField37.setText(dbc.rst.getString(8));

}

}

catch (Exception e)

{

e.printStackTrace();

}

}

**Coding On Update Button: -**

private void jButton23ActionPerformed(java.awt.event.ActionEvent evt) {

String user2=jTextField40.getText();

String name=jTextField39.getText();

String address=jTextArea6.getText();

String Mobile=jTextField37.getText();

if(jRadioButton11.isSelected())

{

gender="Male";

}

if(jRadioButton12.isSelected())

{

gender="Female";

}

try

{

DB db=new DB();

db.pstmt=db.con.prepareStatement("update add\_customer set Name=?,Gender=?,Address=?,Mobile=? where Customer\_ID='"+user2+"' ");

db.pstmt.setString(1, name);

db.pstmt.setString(2, gender);

db.pstmt.setString(3, address);

db.pstmt.setString(4, Mobile);

int i=db.pstmt.executeUpdate();

if(i>0)

{

JOptionPane.showMessageDialog(this, "Customer Record Updated Successfully");

jTextField40.setText(null);

jTextField39.setText(null);

jTextArea6.setText(null);

jTextField37.setText(null);

jRadioButton11.setSelected(false);

jRadioButton12.setSelected(false);

}

else

{

JOptionPane.showMessageDialog(this, "ERROR!!!!!!!!!!");

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Coding On Delete Button: -**

private void jButton24ActionPerformed(java.awt.event.ActionEvent evt) {

try

{

String user2=jTextField40.getText();

DB db=new DB();

db.pstmt=db.con.prepareStatement("delete from add\_Customer where Customer\_ID='"+user2+"' ");

int i=db.pstmt.executeUpdate();

if(i>0)

{

JOptionPane.showMessageDialog(this, "Driver Record Deleted Successfully");

jTextField40.setText(null);

jTextField39.setText(null);

jTextArea6.setText(null);

jTextField37.setText(null);

jRadioButton11.setSelected(false);

jRadioButton12.setSelected(false);

}

else

{

JOptionPane.showMessageDialog(this, "Error!!!");

}

}

catch (Exception e)

{

e.printStackTrace();

}

}

**Return Car: -**

**Event Performed On Menu Item:-**

Action Event performed(actionPerformed)

**Coding: -**

private void jMenuItem14ActionPerformed(java.awt.event.ActionEvent evt)

{

Return\_Car.setVisible(true);

}

**Coding On Search By ID Button: -**

private void jButton26ActionPerformed(java.awt.event.ActionEvent evt)

{

String user2=jTextField38.getText();

try

{

DB dbc=new DB();

dbc.pstmt=dbc.con.prepareStatement("select \* from add\_customer where Customer\_ID='"+user2+"'");

dbc.rst=dbc.pstmt.executeQuery();

if(dbc.rst.next())

{

jTextField41.setText(dbc.rst.getString(2));

jTextField42.setText(dbc.rst.getString(11));

jTextField44.setText(dbc.rst.getString(9));

jTextField47.setText(dbc.rst.getString(6));

jTextField45.setText(dbc.rst.getString(7));

}

}

catch (Exception e)

{

e.printStackTrace();

}

String car\_number=jTextField42.getText();

try

{

DB dbc=new DB();

dbc.pstmt=dbc.con.prepareStatement("select Model from add\_vehicle where Car\_number='"+car\_number+"'");

dbc.rst=dbc.pstmt.executeQuery();

if(dbc.rst.next())

{

jTextField43.setText(dbc.rst.getString(1));

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Coding On Search By Name Button: -**

private void jButton27ActionPerformed(java.awt.event.ActionEvent evt) {

String user2=jTextField41.getText();

try

{

DB dbc=new DB();

dbc.pstmt=dbc.con.prepareStatement("select \* from add\_customer where Name='"+user2+"'");

dbc.rst=dbc.pstmt.executeQuery();

if(dbc.rst.next())

{

jTextField38.setText(dbc.rst.getString(1));

jTextField42.setText(dbc.rst.getString(11));

jTextField44.setText(dbc.rst.getString(9));

jTextField47.setText(dbc.rst.getString(6));

jTextField45.setText(dbc.rst.getString(7));

}

}

catch (Exception e)

{

e.printStackTrace();

}

String car\_number=jTextField42.getText();

try

{

DB dbc=new DB();

dbc.pstmt=dbc.con.prepareStatement("select Model from add\_vehicle where Car\_number='"+car\_number+"'");

dbc.rst=dbc.pstmt.executeQuery();

if(dbc.rst.next())

{

jTextField43.setText(dbc.rst.getString(1));

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Coding On Returned Button: -**

private void jButton28ActionPerformed(java.awt.event.ActionEvent evt) {

String customer\_id=jTextField38.getText();

String name=jTextField41.getText();

String car\_number=jTextField42.getText();

String model=jTextField43.getText();

String company=jTextField44.getText();

String Booking\_Date=jTextField47.getText();

String Return\_Date=jTextField45.getText();

try

{

DB db=new DB();

db.pstmt=db.con.prepareStatement("insert into return\_car values(?,?,?,?,?,?,?)");

db.pstmt.setString(1, customer\_id);

db.pstmt.setString(2, name);

db.pstmt.setString(3, car\_number);

db.pstmt.setString(4,model);

db.pstmt.setString(5, company);

db.pstmt.setString(6, Booking\_Date);

db.pstmt.setString(7, Return\_Date);

int i=db.pstmt.executeUpdate();

if(i>0)

{

JOptionPane.showMessageDialog(this, "Vehicle Returned Successfully");

jTextField38.setText(null);

jTextField41.setText(null);

jTextField42.setText(null);

jTextField43.setText(null);

jTextField44.setText(null);

jTextField45.setText(null);

jTextField47.setText(null);

}

else

{

JOptionPane.showMessageDialog(this, "ERROR!!!!!!!!!!");

}

}

catch(Exception e)

{

e.printStackTrace();

}

try

{

DB db=new DB();

db.pstmt=db.con.prepareStatement("Update Avaliability from add\_vehicle where Car\_number='"+car\_number+"' ");

db.pstmt.setString(1,"1");

db.pstmt.executeUpdate();

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Coding On Cancel Button: -**

private void jButton29ActionPerformed(java.awt.event.ActionEvent evt)

{

Return\_Car.setVisible(false);

}

**View Cars: -**

**Event Performed On Menu Item:-**

Action Event performed(actionPerformed)

**Coding: -**

private void jMenuItem6ActionPerformed(java.awt.event.ActionEvent evt)

{

View\_Cars.setVisible(true);

try

{

DB db=new DB();

db.pstmt=db.con.prepareStatement("select Company\_name from add\_company");

db.rst=db.pstmt.executeQuery();

while(db.rst.next())

{

jComboBox9.addItem(db.rst.getString(1));

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Coding On View Available Cars: -**

private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {

try

{

String company=(String)jComboBox9.getSelectedItem();

String Use\_as=(String)jComboBox11.getSelectedItem();

String avail;

String a="select \* from add\_vehicle where Avaliability=1 AND Company='"+company+"' ";

String b="select \* from add\_vehicle where Avaliability=0 AND Company='"+company+"' ";

String q="select \* from add\_vehicle where Avaliability=0 AND Company='"+company+"' ";

avail=(String)jComboBox10.getSelectedItem();

if(avail.equals("Available"))

{

q=a;

}

if(avail.equals("Non Avaliable"))

{

q=b;

}

Vector<Vector<String>>data=new Vector<Vector<String>>();

data.removeAllElements();

Vector<String>header=new Vector<String>();

header.removeAllElements();

header.add("Model");

header.add("Car Number");

header.add("Make Year");

header.add("Seating Capacity");

header.add("Type");

header.add("Varient");

header.add("Engine Capcity");

header.add("Rent");

DB db=new DB();

db.stmt=db.con.createStatement();

db.rst=db.stmt.executeQuery(q);

while(db.rst.next())

{

Vector<String>data1=new Vector<String>();

data1.add(db.rst.getString(1));

data1.add(db.rst.getString(3));

data1.add(db.rst.getString(4));

data1.add(db.rst.getString(6));

data1.add(db.rst.getString(7));

data1.add(db.rst.getString(8));

data1.add(db.rst.getString(9));

data1.add(db.rst.getString(10));

data.add(data1);

jTable3.setModel(new javax.swing.table.DefaultTableModel(data,header));

}

}

catch(Exception e)

{

e.printStackTrace(); } }

**Change Password: -**

**Event Performed On Menu Item:-**

Action Event performed(actionPerformed)

**Coding: -**

private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt)

{

Change\_Password.setVisible(true);

}

**Coding On Submit Button: -**

private void jButton12ActionPerformed(java.awt.event.ActionEvent evt)

{

try

{

String user1=user.getText();

String password=pf2.getText();

int a=1;

String message="";

DB dbc=new DB();

String newpassword=tg2.getText();

String conpassword=tg3.getText();

if(user1.equals(""))

{

message=message+"User Field Can not be Empty ";

a=0;

}

if(password.equals(""))

{

message=message+" Password Field can not be Empty ";

a=0;

}

if(newpassword.equals(""))

{

message=message+" New Password Field can not be Empty ";

a=0;

}

if(conpassword.equals(""))

{

message=message+" Confirm Password Field can not be Empty ";

a=0;

}

if(a==0)

{

JOptionPane.showMessageDialog(this, message);

}

if(a==1)

{

dbc.pstmt=dbc.con.prepareStatement("select Password from driver\_login where password=?");

dbc.pstmt.setString(1,password);

dbc.rst=dbc.pstmt.executeQuery();

if(dbc.rst.next())

{

if(newpassword.equals(conpassword))

{

dbc.pstmt=dbc.con.prepareStatement("update driver\_login set Password=? where user='"+user1+"' ");

dbc.pstmt.setString(1,conpassword);

int i=dbc.pstmt.executeUpdate();

if(i>0)

{

JOptionPane.showMessageDialog(this,"Changed Successfully");

}

else

{

JOptionPane.showMessageDialog(this,"Error in changing Password!!");

}

}

else

{

JOptionPane.showMessageDialog(this,"Password Not Matched Please Try again");

}

}

else

{

JOptionPane.showMessageDialog(this,"Incorrect Password");

}

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Coding On Cancel Button: -**

private void jButton13ActionPerformed(java.awt.event.ActionEvent evt)

{

Change\_Password.setVisible(false);

}

**Admin Main Window: -**

**Add Employee: -**

**Event Performed On Menu Item:-**

Action Event performed(actionPerformed)

**Coding: -**

private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt)

{

Add\_Employee.setVisible(true);

}

**Coding On Submit Button: -**

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)

{

try

{

String name=tt2.getText();

String address=ta1.getText();

String DOB=tt3.getText();

String Qualification=(String) c2.getSelectedItem();

String Mobile=tt4.getText();

String user2=tt5.getText();

String password=tt6.getText();

String conpassword=tt7.getText();

String DOJ=tt8.getText();

if(r1.isSelected())

{

gender="Male";

}

if(r2.isSelected())

{

gender="Female";

}

int a=1;

String message="";

if(name.equals(""))

{

message=message+"Name Can not be Empty ";

a=0;

}

if(address.equals(""))

{

message=message+" Address can not be Empty ";

a=0;

}

if(DOB.equals(""))

{

message=message+" DOB can not be Empty ";

a=0;

}

if(Mobile.equals(""))

{

message=message+" Mobile can not be Empty ";

a=0;

}

if(user2.equals(""))

{

message=message+" User Name can not be Empty ";

a=0;

}

if(password.equals(""))

{

message=message+" Password Field can not be Empty ";

a=0;

}

if(conpassword.equals(""))

{

message=message+" Confirm Password Field can not be Empty ";

a=0;

}

if(DOJ.equals(""))

{

message=message+" Date Of Joinging can not be Empty ";

a=0;

}

if(a==0)

{

JOptionPane.showMessageDialog(this, message);

}

if(a==1)

{

if(password.equals(conpassword))

{

DB db=new DB();

db.pstmt=db.con.prepareStatement("insert into add\_emp values(?,?,?,?,?,?,?,?,?)");

db.pstmt.setString(1, name);

db.pstmt.setString(2, gender);

db.pstmt.setString(3, address);

db.pstmt.setString(4,DOB);

db.pstmt.setString(5, Qualification);

db.pstmt.setString(6, Mobile);

db.pstmt.setString(7, user2);

db.pstmt.setString(8, password);

db.pstmt.setString(9, DOJ);

int i=db.pstmt.executeUpdate();

if(i>0)

{

JOptionPane.showMessageDialog(this, "Employee Added Successfully");

tt2.setText(null);

ta1.setText(null);

tt3.setText(null);

c2.setSelectedItem("Select");

tt4.setText(null);

tt5.setText(null);

tt6.setText(null);

tt7.setText(null);

tt8.setText(null);

r1.setSelected(false);

r2.setSelected(false);

}

else

{

JOptionPane.showMessageDialog(this, "ERROR!!!!!!!!!!");

}

}

else

{

JOptionPane.showMessageDialog(this,"Password Not Matched Please Try again");

}

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Coding On Cancel Button: -**

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)

{

Add\_Employee.setVisible(false);

}

**Modify Employee: -**

**Event Performed On Menu Item:-**

Action Event performed(actionPerformed)

**Coding: -**

private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt)

{

Modify\_Employee.setVisible(true);

}

**Coding On Search Button: -**

private void jButton10ActionPerformed(java.awt.event.ActionEvent evt) {

String user2=jTextField11.getText();

try {

DB dbc=new DB();

dbc.pstmt=dbc.con.prepareStatement("select \* from add\_emp where user='"+user2+"'");

dbc.rst=dbc.pstmt.executeQuery();

if(dbc.rst.next())

{

jTextField8.setText(dbc.rst.getString(1));

String s=dbc.rst.getString(2);

if(s.equals("Male"))

{

jRadioButton3.setSelected(true);

jRadioButton4.setSelected(false);

}

if(s.equals("Female"))

{

jRadioButton4.setSelected(true);

jRadioButton3.setSelected(false);

}

jTextArea2.setText(dbc.rst.getString(3));

jTextField9.setText(dbc.rst.getString(4));

String q=dbc.rst.getString(5);

for(int h=0;h<jComboBox2.getItemCount();h++)

{

if(q.equals(jComboBox2.getItemAt(h)))

{

jComboBox2.setSelectedIndex(h);

}

}

jTextField10.setText(dbc.rst.getString(6));

jTextField12.setText(dbc.rst.getString(9));

}

} catch (Exception e) {

e.printStackTrace();

}

}

**Coding On Submit Button: -**

private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {

String user2=jTextField11.getText();

String name=jTextField8.getText();

String address=jTextArea2.getText();

String DOB=jTextField9.getText();

String qualification=(String) jComboBox2.getSelectedItem();

String Mobile=jTextField10.getText();

String DOJ=jTextField12.getText();

if(jRadioButton3.isSelected())

{

gender="Male";

}

if(jRadioButton4.isSelected())

{

gender="Female";

}

try

{

DB db=new DB();

db.pstmt=db.con.prepareStatement("update add\_emp set Name=?,Gender=?,Address=?,DOB=?,Qualification=?,Mobile=?,Date\_of\_Joining=? where user='"+user2+"' ");

db.pstmt.setString(1, name);

db.pstmt.setString(2, gender);

db.pstmt.setString(3, address);

db.pstmt.setString(4,DOB);

db.pstmt.setString(5, qualification);

db.pstmt.setString(6, Mobile);

db.pstmt.setString(7, DOJ);

int i=db.pstmt.executeUpdate();

if(i>0)

{

JOptionPane.showMessageDialog(this, "Employee Record Updated Successfully");

jTextField11.setText(null);

jTextField8.setText(null);

jTextArea2.setText(null);

jComboBox2.setSelectedItem("Select");

jTextField9.setText(null);

jTextField10.setText(null);

jTextField12.setText(null);

jRadioButton3.setSelected(false);

jRadioButton4.setSelected(false);

}

else

{

JOptionPane.showMessageDialog(this, "ERROR!!!!!!!!!!");

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Coding On Cancel Button: -**

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt)

{

Modify\_Employee.setVisible(false);

}

**Employee List: -**

**Event Performed On Menu Item:-**

Action Event performed(actionPerformed)

**Coding: -**

private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {

Employees\_List.setVisible(true);

try

{

Vector<Vector<String>>data=new Vector<Vector<String>>();

data.removeAllElements();

Vector<String>header=new Vector<String>();

header.removeAllElements();

header.add("Name");

header.add("Gender");

header.add("Address");

header.add("DOB");

header.add("Qualification");

header.add("Mobile");

header.add("User Name");

header.add("Date Of Joining");

DB db=new DB();

db.stmt=db.con.createStatement();

db.rst=db.stmt.executeQuery("select \* from add\_emp");

while(db.rst.next())

{

Vector<String>data1=new Vector<String>();

data1.add(db.rst.getString(1));

data1.add(db.rst.getString(2));

data1.add(db.rst.getString(3));

data1.add(db.rst.getString(4));

data1.add(db.rst.getString(5));

data1.add(db.rst.getString(6));

data1.add(db.rst.getString(7));

data1.add(db.rst.getString(9));

data.add(data1);

jTable1.setModel(new javax.swing.table.DefaultTableModel(data,header));

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Driver List: -**

**Event Performed On Menu Item:-**

Action Event performed(actionPerformed)

**Coding: -**

private void jMenuItem5ActionPerformed(java.awt.event.ActionEvent evt) {

Drivers\_List.setVisible(true);

try

{

Vector<Vector<String>>data=new Vector<Vector<String>>();

data.removeAllElements();

Vector<String>header=new Vector<String>();

header.removeAllElements();

header.add("Name");

header.add("Gender");

header.add("Address");

header.add("Experience");

header.add("DOB");

header.add("Mobile");

header.add("Driver ID");

header.add("Seating Capacity");

header.add("Company");

header.add("Car Number");

header.add("Driver License");

DB db=new DB();

db.stmt=db.con.createStatement();

db.rst=db.stmt.executeQuery("select \* from add\_Driver");

while(db.rst.next())

{

Vector<String>data1=new Vector<String>();

data1.add(db.rst.getString(1));

data1.add(db.rst.getString(2));

data1.add(db.rst.getString(3));

data1.add(db.rst.getString(4));

data1.add(db.rst.getString(5));

data1.add(db.rst.getString(6));

data1.add(db.rst.getString(7));

data1.add(db.rst.getString(8));

data1.add(db.rst.getString(9));

data1.add(db.rst.getString(10));

data1.add(db.rst.getString(11));

data.add(data1);

jTable2.setModel(new javax.swing.table.DefaultTableModel(data,header));

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**View Cars: -**

**Event Performed On Menu Item:-**

Action Event performed(actionPerformed)

**Coding: -**

private void jMenuItem6ActionPerformed(java.awt.event.ActionEvent evt)

{

View\_Cars.setVisible(true);

try

{

DB db=new DB();

db.pstmt=db.con.prepareStatement("select Company\_name from add\_company");

db.rst=db.pstmt.executeQuery();

while(db.rst.next())

{

jComboBox9.addItem(db.rst.getString(1));

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Coding On View Available Cars Button: -**

private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {

try

{

String company=(String)jComboBox9.getSelectedItem();

String Use\_as=(String)jComboBox11.getSelectedItem();

String avail;

String a="select \* from add\_vehicle where Avaliability=1 AND Company='"+company+"' ";

String b="select \* from add\_vehicle where Avaliability=0 AND Company='"+company+"' ";

String q="select \* from add\_vehicle where Avaliability=0 AND Company='"+company+"' ";

avail=(String)jComboBox10.getSelectedItem();

if(avail.equals("Available"))

{

q=a;

}

if(avail.equals("Non Avaliable"))

{

q=b;

}

Vector<Vector<String>>data=new Vector<Vector<String>>();

data.removeAllElements();

Vector<String>header=new Vector<String>();

header.removeAllElements();

header.add("Model");

header.add("Car Number");

header.add("Make Year");

header.add("Seating Capacity");

header.add("Type");

header.add("Varient");

header.add("Engine Capcity");

header.add("Rent");

DB db=new DB();

db.stmt=db.con.createStatement();

db.rst=db.stmt.executeQuery(q);

while(db.rst.next())

{

Vector<String>data1=new Vector<String>();

data1.add(db.rst.getString(1));

data1.add(db.rst.getString(3));

data1.add(db.rst.getString(4));

data1.add(db.rst.getString(6));

data1.add(db.rst.getString(7));

data1.add(db.rst.getString(8));

data1.add(db.rst.getString(9));

data1.add(db.rst.getString(10));

data.add(data1);

jTable3.setModel(new javax.swing.table.DefaultTableModel(data,header));

}

}

catch(Exception e)

{

e.printStackTrace(); } }

**Add Company: -**

**Event Performed On Menu Item:-**

Action Event performed(actionPerformed)

**Coding: -**

private void jMenuItem7ActionPerformed(java.awt.event.ActionEvent evt) {

Add\_Company.setVisible(true);

try

{

DB db=new DB();

db.pstmt=db.con.prepareStatement("select Company\_name from add\_company");

db.rst=db.pstmt.executeQuery();

while(db.rst.next())

{

cb1.addItem(db.rst.getString(1));

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Coding On Submit Button: -**

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {

try

{

String company=tt1.getText();

DB db=new DB();

db.pstmt=db.con.prepareStatement("insert into add\_company values(?)");

db.pstmt.setString(1, company);

int i=db.pstmt.executeUpdate();

if(i>0)

{

JOptionPane.showMessageDialog(this, "Company Added Successfully");

tt1.setText(null);

}

else

{

JOptionPane.showMessageDialog(this, "ERROR!!!!!!!!!!");

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Coding On Cancel Button: -**

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt)

{

Add\_Company.setVisible(false);

}

**ADD Vehicle: -**

**Event Performed On Menu Item:-**

Action Event performed(actionPerformed)

**Coding: -**

private void jMenuItem13ActionPerformed(java.awt.event.ActionEvent evt) {

try

{

Add\_Vehicle.setVisible(true);

DB db=new DB();

db.pstmt=db.con.prepareStatement("select Company\_name from add\_company");

db.rst=db.pstmt.executeQuery();

while(db.rst.next())

{

jComboBox4.addItem(db.rst.getString(1));

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Coding On Submit Button: -**

private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {

try

{

String avaliablity="1";

String model=jTextField14.getText();

String company=(String) jComboBox4.getSelectedItem();

String car\_number=jTextField15.getText();

String make\_year=jTextField16.getText();

String use\_as=(String) jComboBox5.getSelectedItem();

String seating\_capacity=(String) jComboBox6.getSelectedItem();

String type=(String) jComboBox7.getSelectedItem();

String varient=(String) jComboBox8.getSelectedItem();

String engine\_capacity=jTextField17.getText();

String rent=jTextField18.getText();

int a=1;

String message="";

if(model.equals(""))

{

message=message+"Model Can not be Empty ";

a=0;

}

if(car\_number.equals(""))

{

message=message+" Car Number can not be Empty ";

a=0;

}

if(make\_year.equals(""))

{

message=message+" Make Year can not be Empty ";

a=0;

}

if(engine\_capacity.equals(""))

{

message=message+" Engine Capacity can not be Empty ";

a=0;

}

if(rent.equals(""))

{

message=message+" Rent Field can not be Empty ";

a=0;

}

if(a==0)

{

JOptionPane.showMessageDialog(this, message);

}

if(a==1)

{

DB db=new DB();

db.pstmt=db.con.prepareStatement("insert into add\_vehicle values(?,?,?,?,?,?,?,?,?,?,?)");

db.pstmt.setString(1, model);

db.pstmt.setString(2, company);

db.pstmt.setString(3, car\_number);

db.pstmt.setString(4,make\_year);

db.pstmt.setString(5, use\_as);

db.pstmt.setString(6, seating\_capacity);

db.pstmt.setString(7, type);

db.pstmt.setString(8, varient);

db.pstmt.setString(9, engine\_capacity);

db.pstmt.setString(10, rent);

db.pstmt.setString(11,avaliablity);

int i=db.pstmt.executeUpdate();

if(i>0)

{

JOptionPane.showMessageDialog(this, "Vehicle Added Successfully");

jTextField14.setText(null);

jTextField15.setText(null);

jTextField16.setText(null);

jComboBox4.setSelectedItem("Select");

jComboBox5.setSelectedItem("Select");

jComboBox6.setSelectedItem("Select");

jComboBox7.setSelectedItem("Select");

jComboBox8.setSelectedItem("Select");

jTextField17.setText(null);

jTextField18.setText(null);

}

else

{

JOptionPane.showMessageDialog(this, "ERROR!!!!!!!!!!");

}

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Coding On Cancel Button: -**

private void jButton8ActionPerformed(java.awt.event.ActionEvent evt)

{

Add\_Vehicle.setVisible(false);

}

**Change Password: -**

**Event Performed On Menu Item:-**

Action Event performed(actionPerformed)

**Coding: -**

private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt)

{

Change\_Password.setVisible(true);

}

**Coding On Submit Button: -**

private void jButton12ActionPerformed(java.awt.event.ActionEvent evt) {

try

{

String user1=user.getText();

String password=pf2.getText();

String newpassword=tg2.getText();

String conpassword=tg3.getText();

int a=1;

String message="";

if(user1.equals(""))

{

message=message+"User Field Can not be Empty ";

a=0;

}

if(password.equals(""))

{

message=message+" Password Field can not be Empty ";

a=0;

}

if(newpassword.equals(""))

{

message=message+" New Password Field can not be Empty ";

a=0;

}

if(conpassword.equals(""))

{

message=message+" Confirm Password Field can not be Empty ";

a=0;

}

if(a==0)

{

JOptionPane.showMessageDialog(this, message);

}

if(a==1)

{

DB dbc=new DB();

dbc.pstmt=dbc.con.prepareStatement("select Password from employee\_login where password=?");

dbc.pstmt.setString(1,password);

dbc.rst=dbc.pstmt.executeQuery();

if(dbc.rst.next())

{

if(newpassword.equals(conpassword))

{

dbc.pstmt=dbc.con.prepareStatement("update employee\_login set Password=? where user='"+user1+"' ");

dbc.pstmt.setString(1,conpassword);

int i=dbc.pstmt.executeUpdate();

if(i>0)

{

JOptionPane.showMessageDialog(this,"Changed Successfully");

}

else

{

JOptionPane.showMessageDialog(this,"Error in changing Password!!");

}

}

else

{

JOptionPane.showMessageDialog(this,"Password Not Matched Please Try again");

}

}

else

{

JOptionPane.showMessageDialog(this,"Incorrect Password");

}

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

**Coding On Cancel Button: -**

private void jButton13ActionPerformed(java.awt.event.ActionEvent evt)

{

Change\_Password.setVisible(false);

}

DESIGN SCREENSHOTS

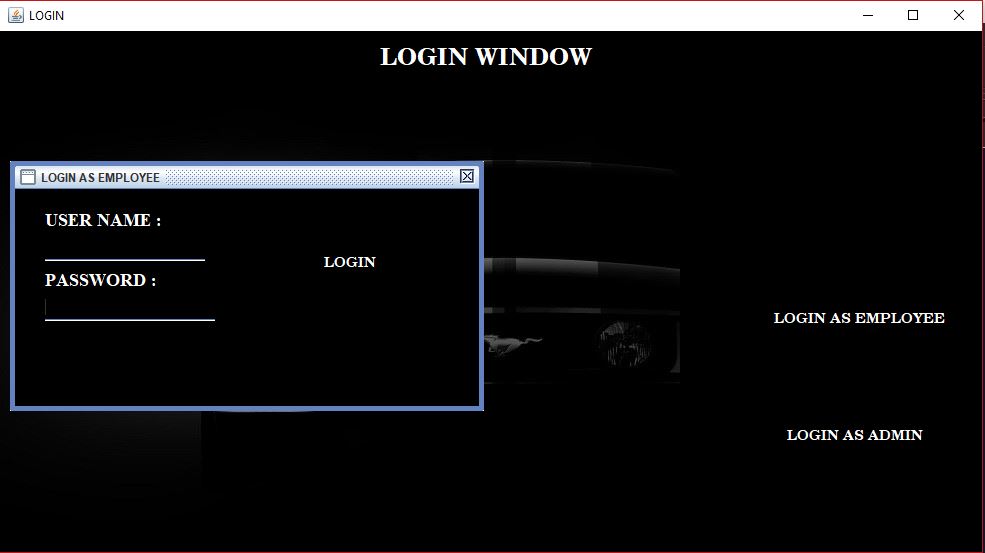
**Splash Screen: -**



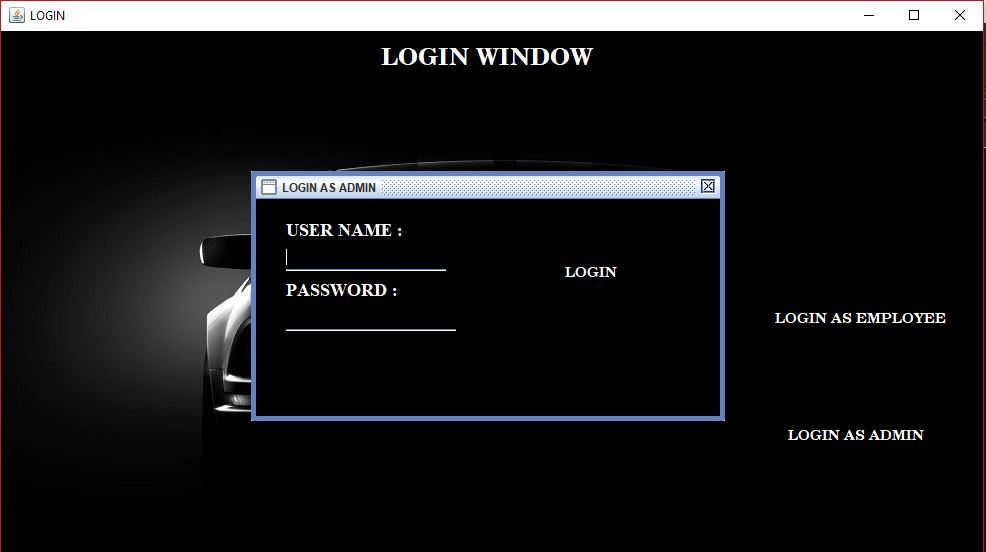
**Login Window: -**



**Employee Login: -**



**Admin Login: -**



**Employee Window: -**



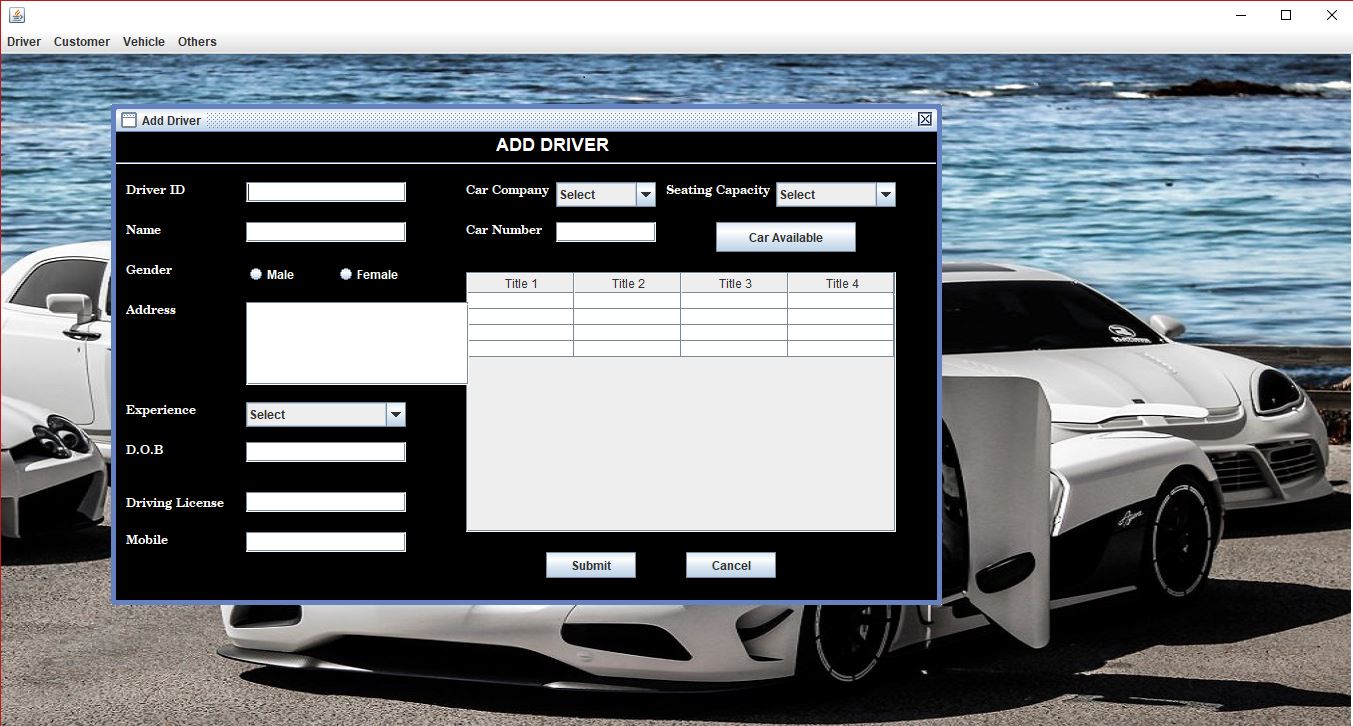




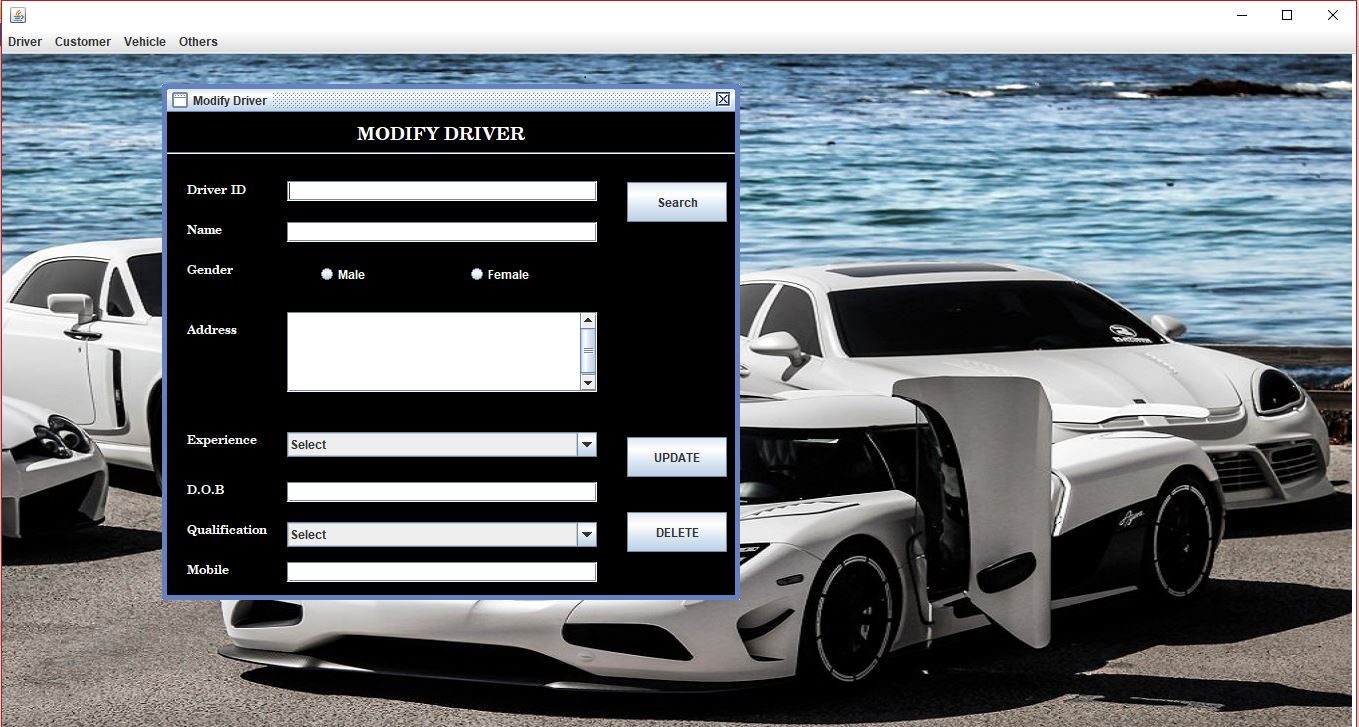




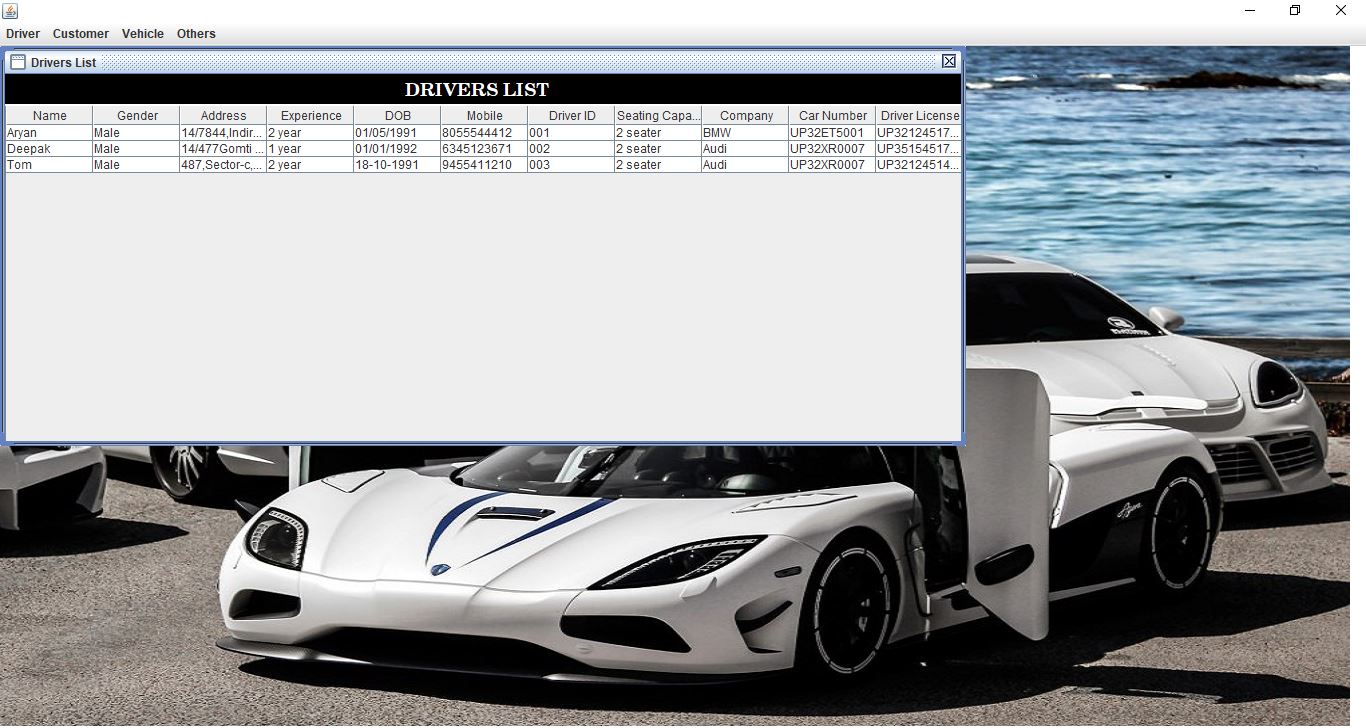
**Add Driver: -**



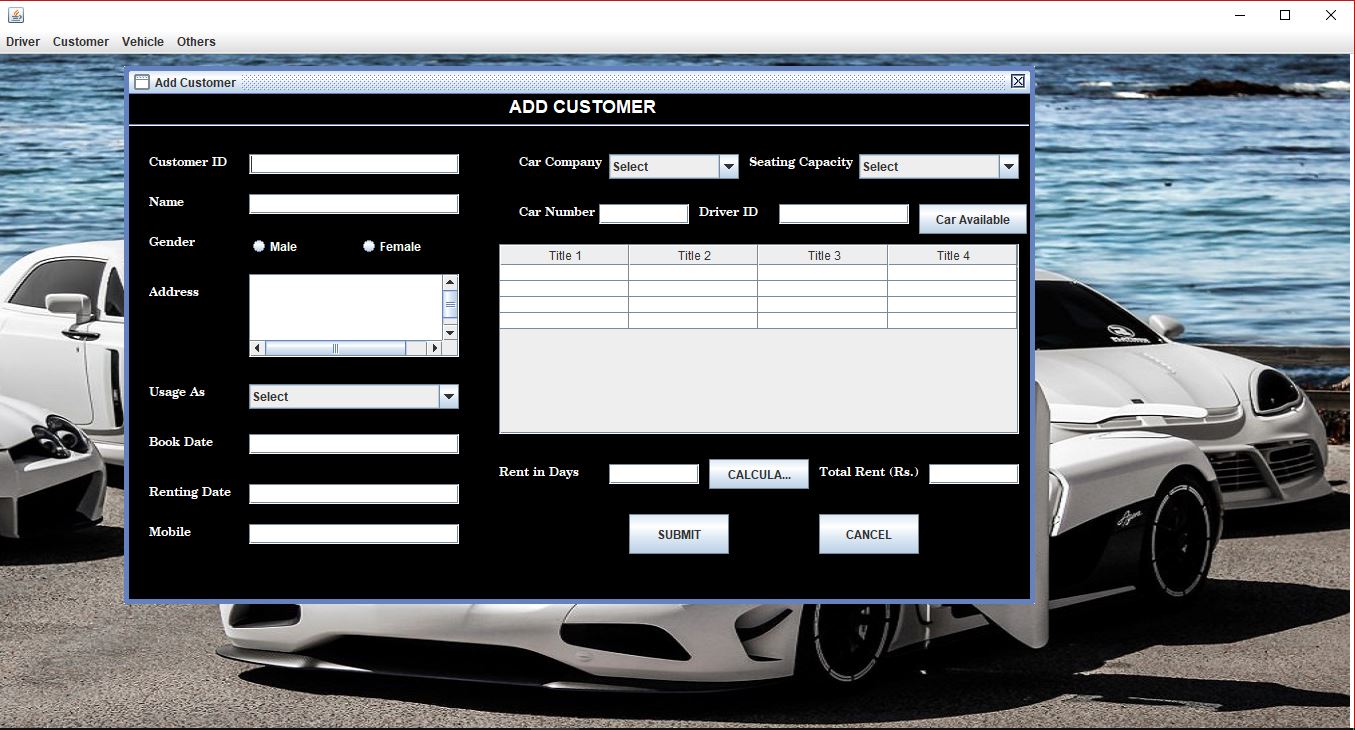
**Modify Driver: -**



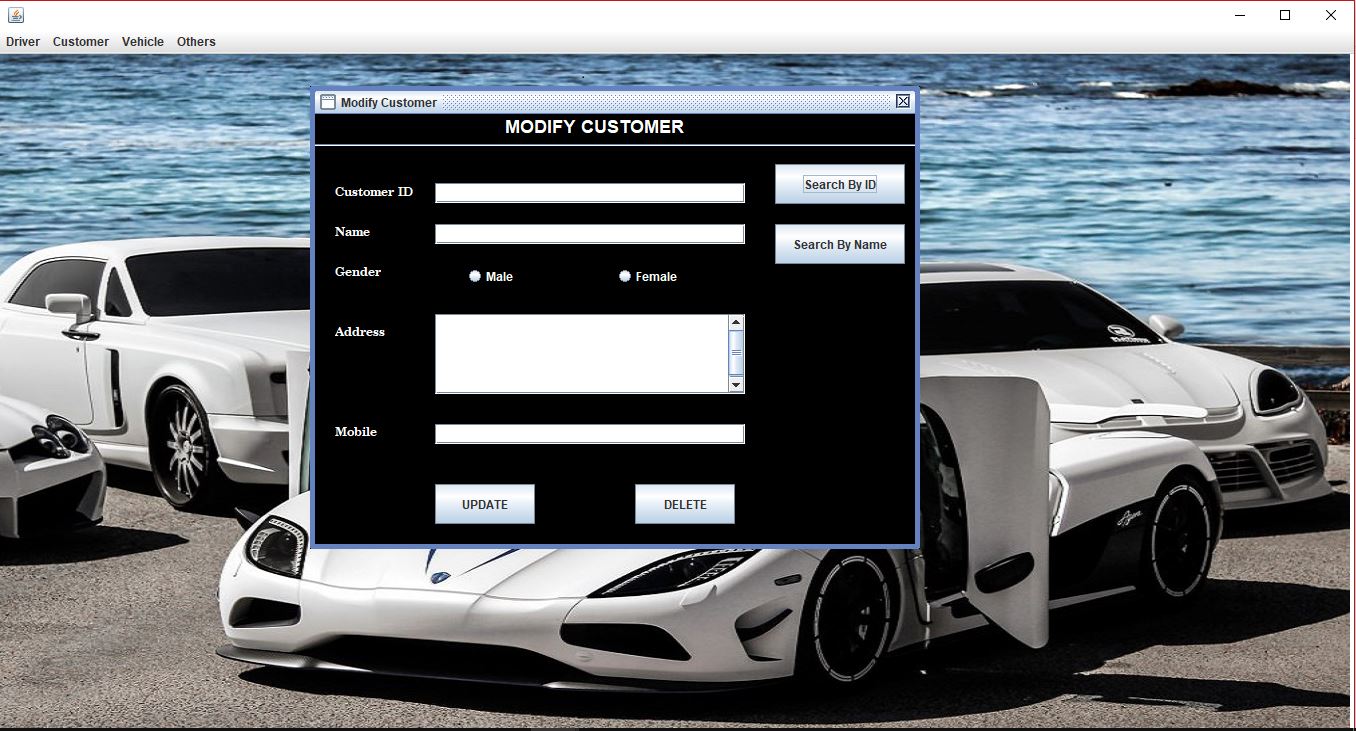
**Driver List: -**



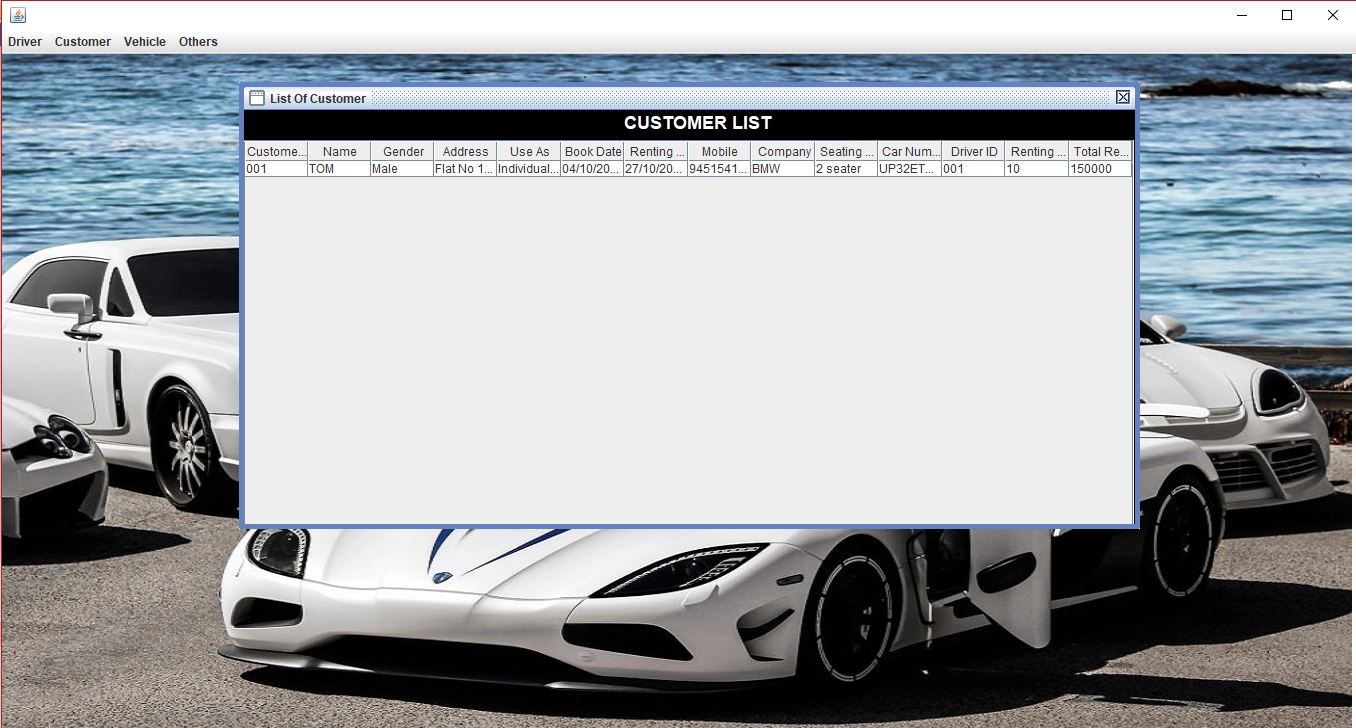
**Add Customer: -**



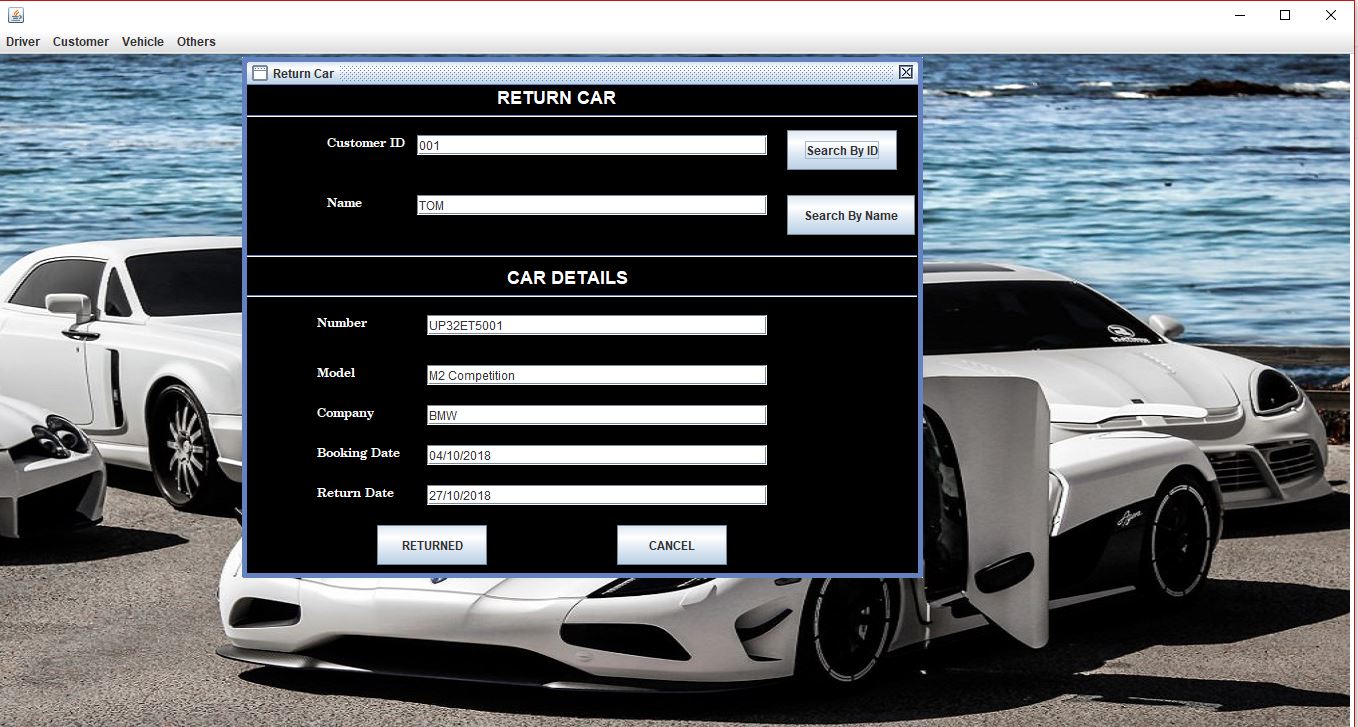
**Modify Customer: -**



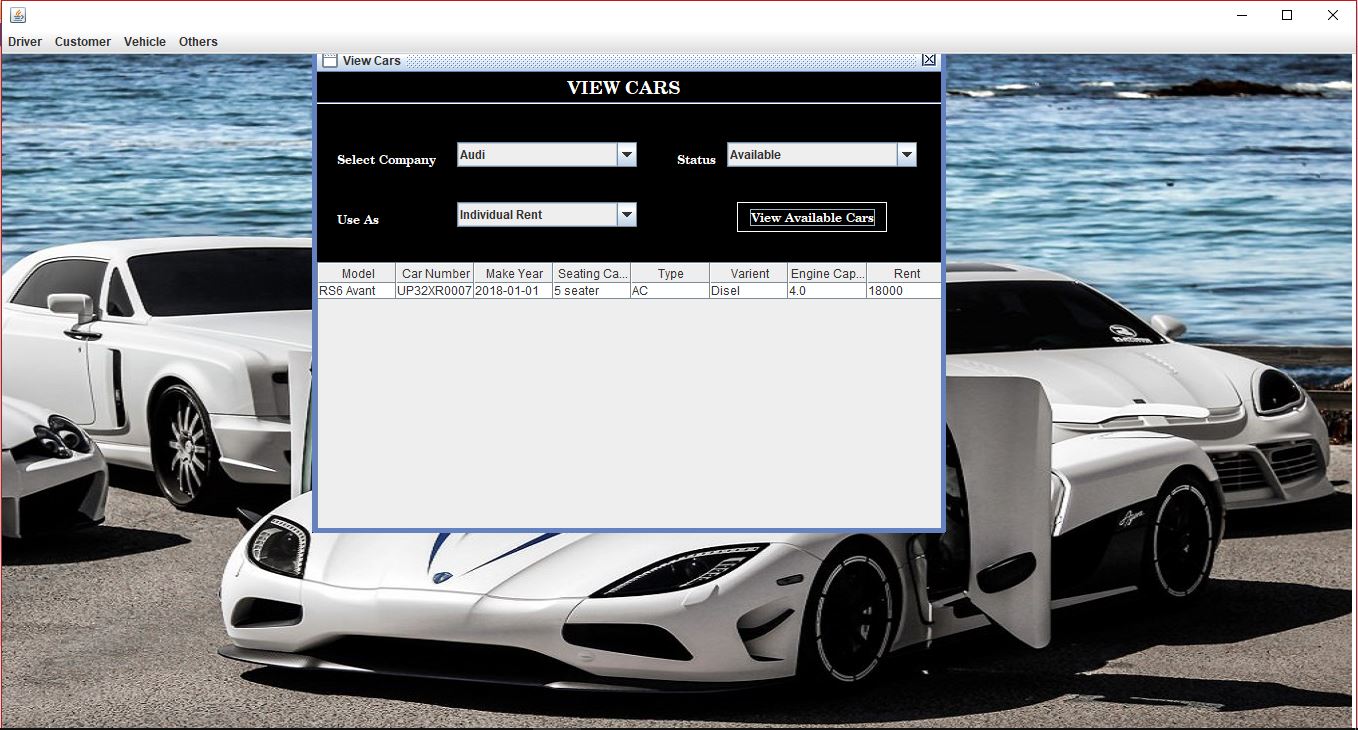
**Customer List: -**



**Return Car: -**



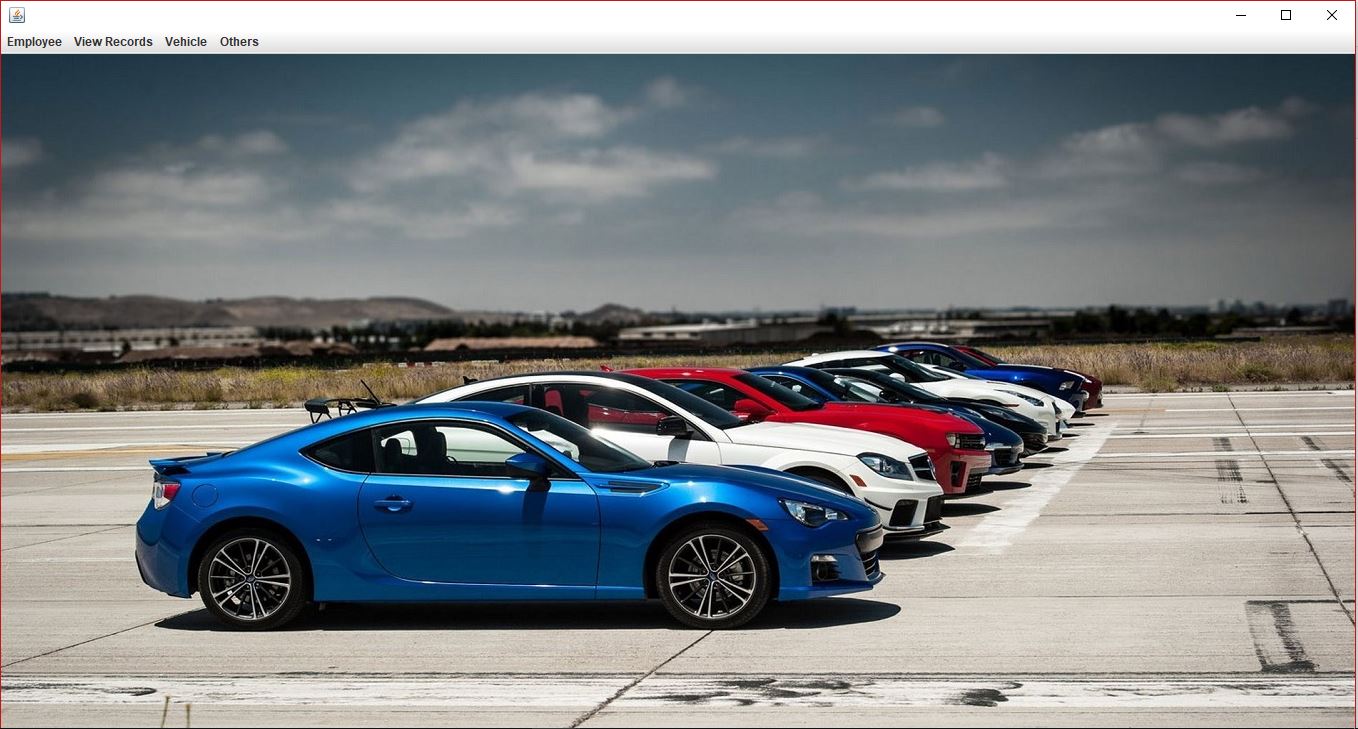
**View Cars: -**

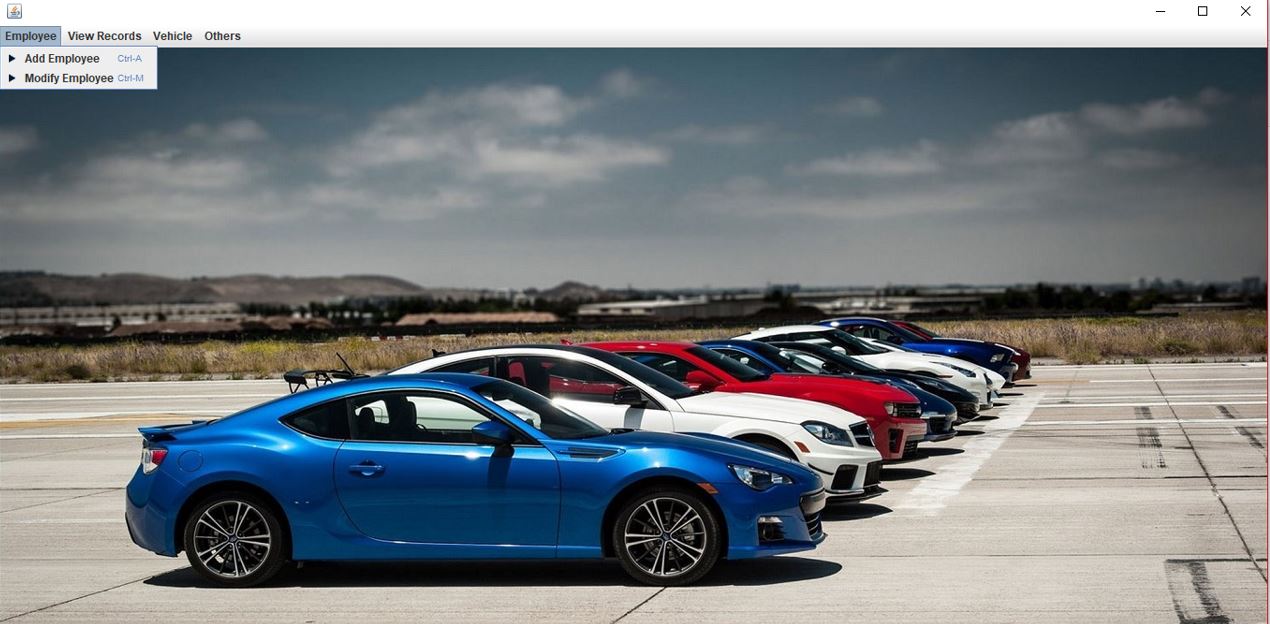


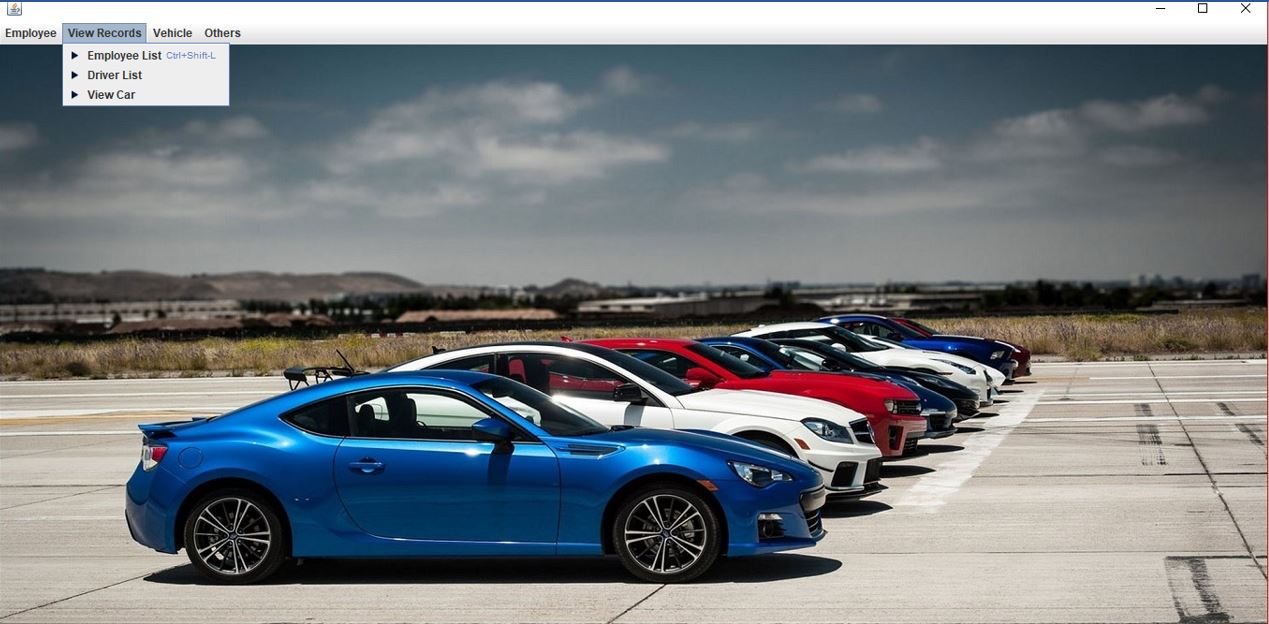
**Change Password: -**



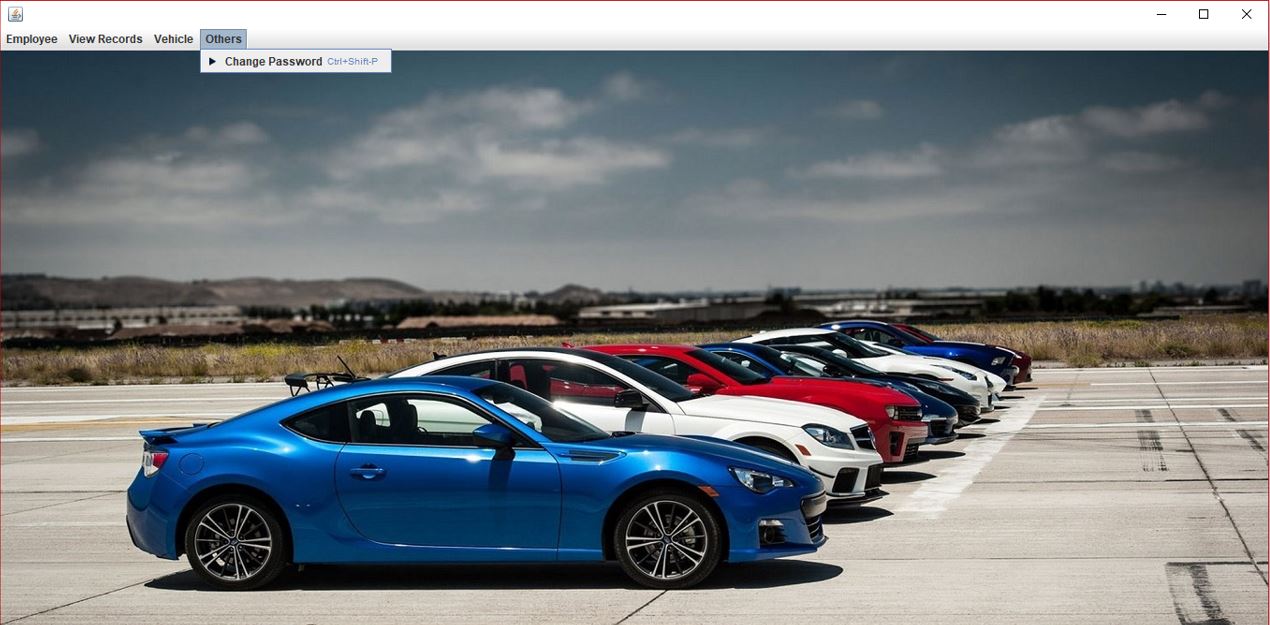
**Admin Window: -**



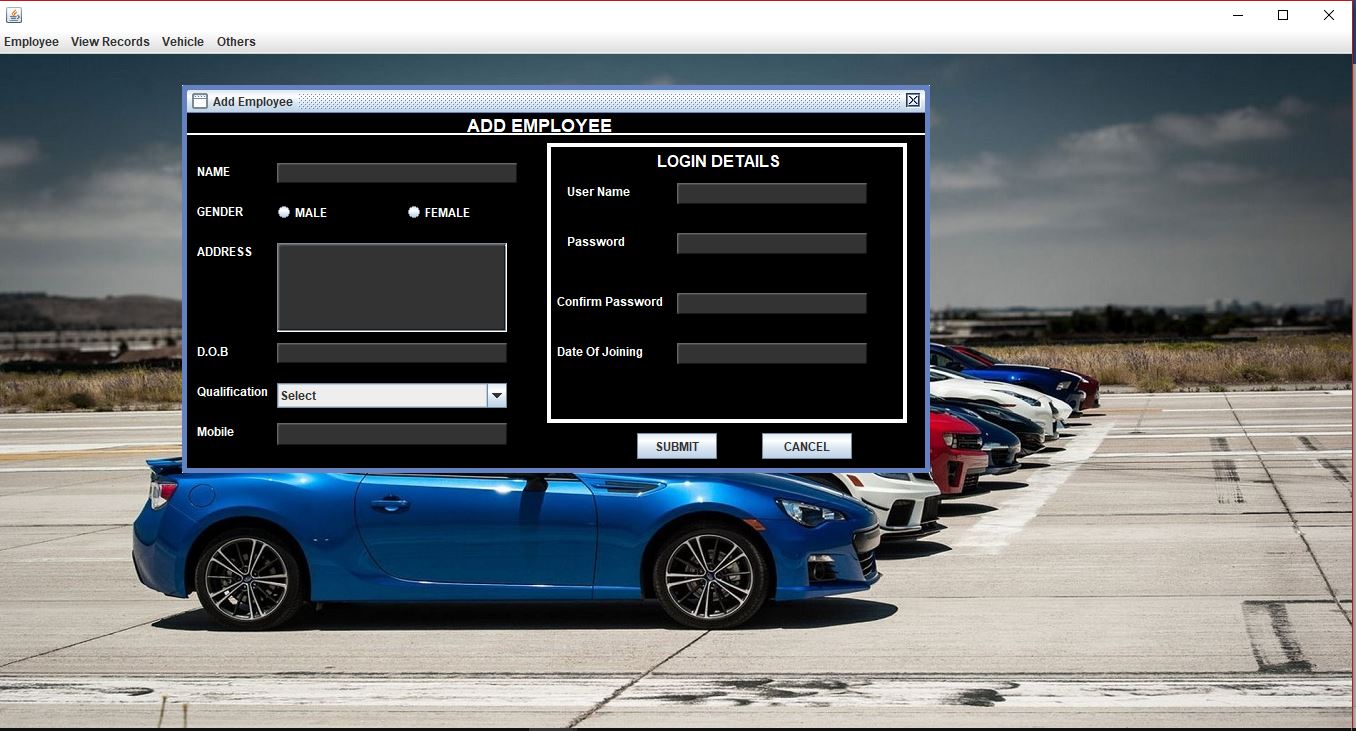




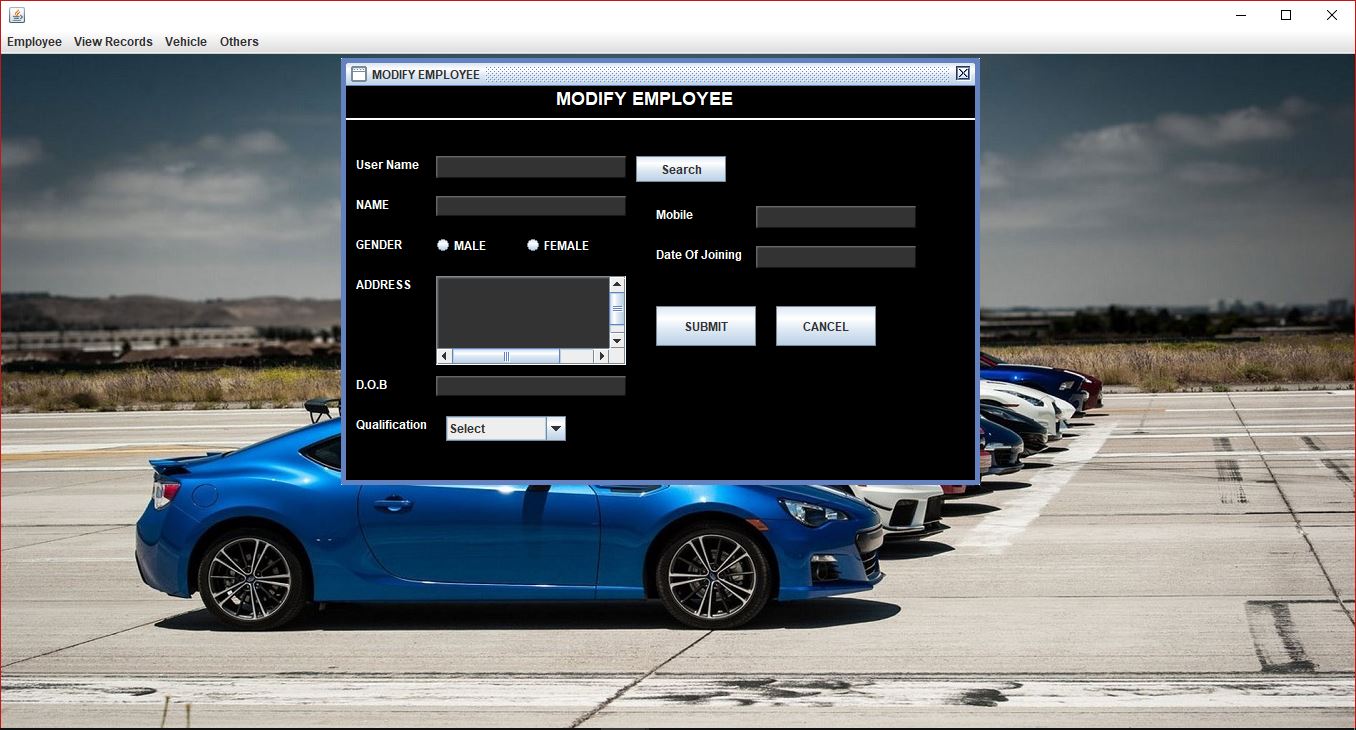




**Add Employee: -**

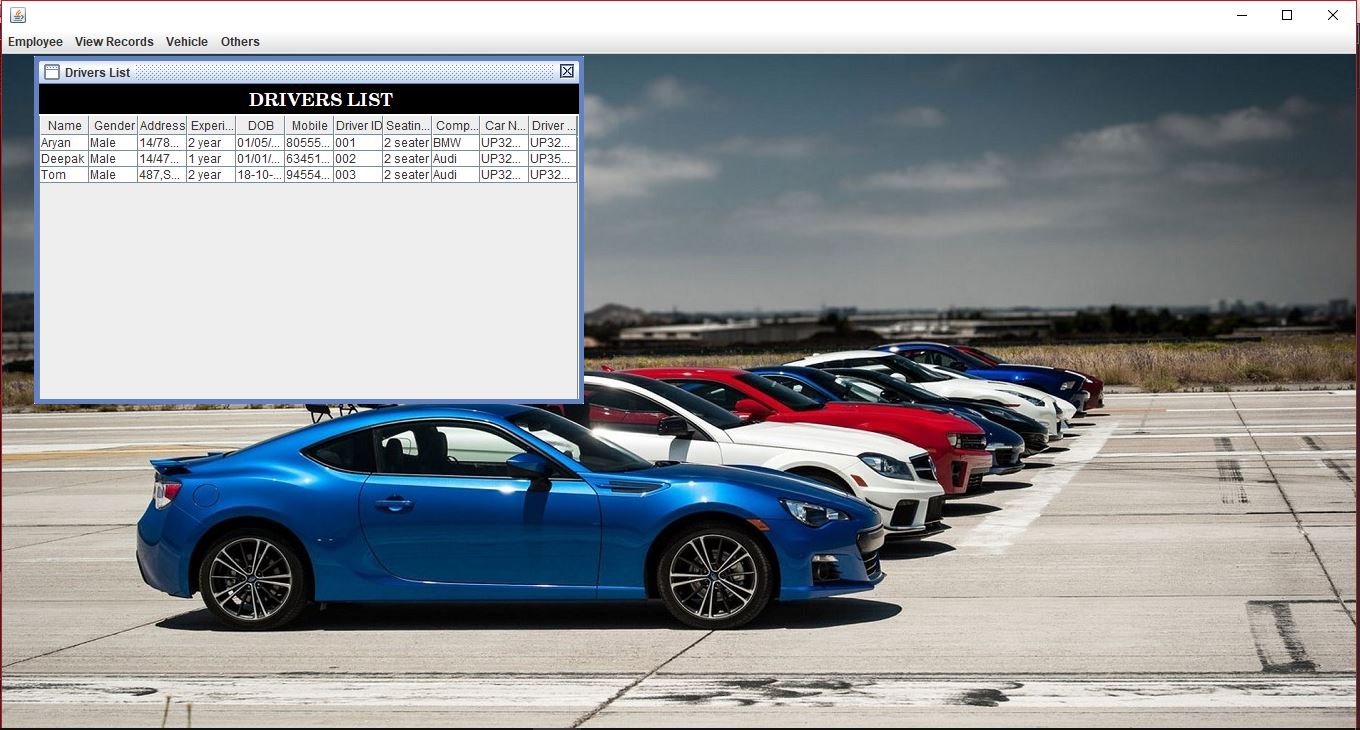


**Modify Employee: -**

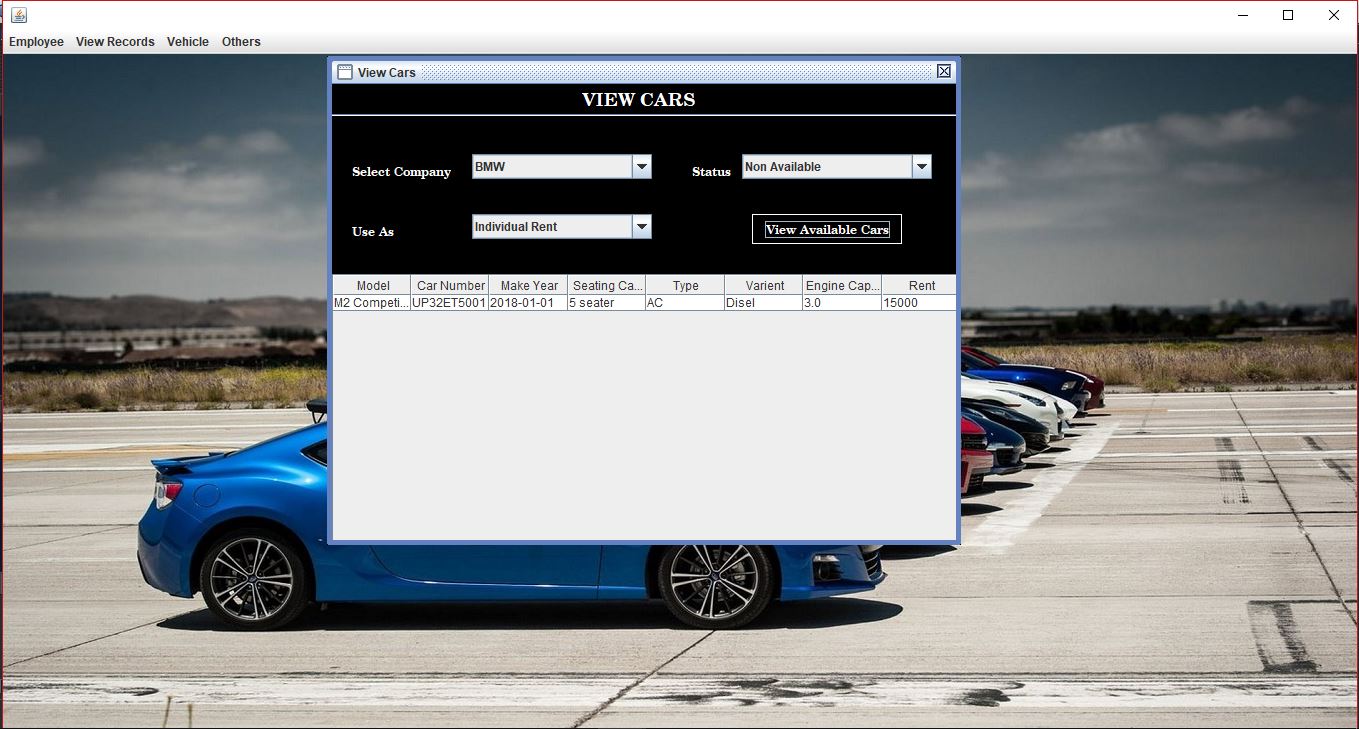


**Employee List: -**

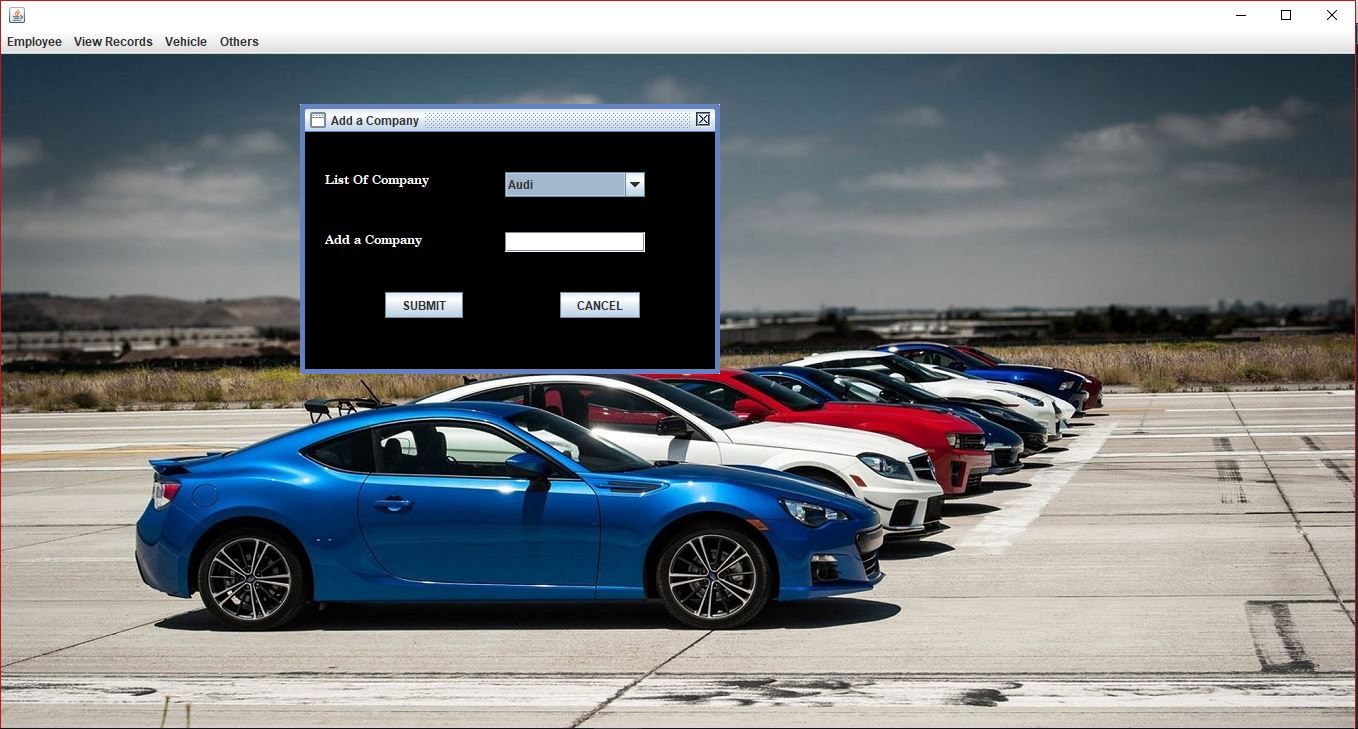
**Driver List: -**



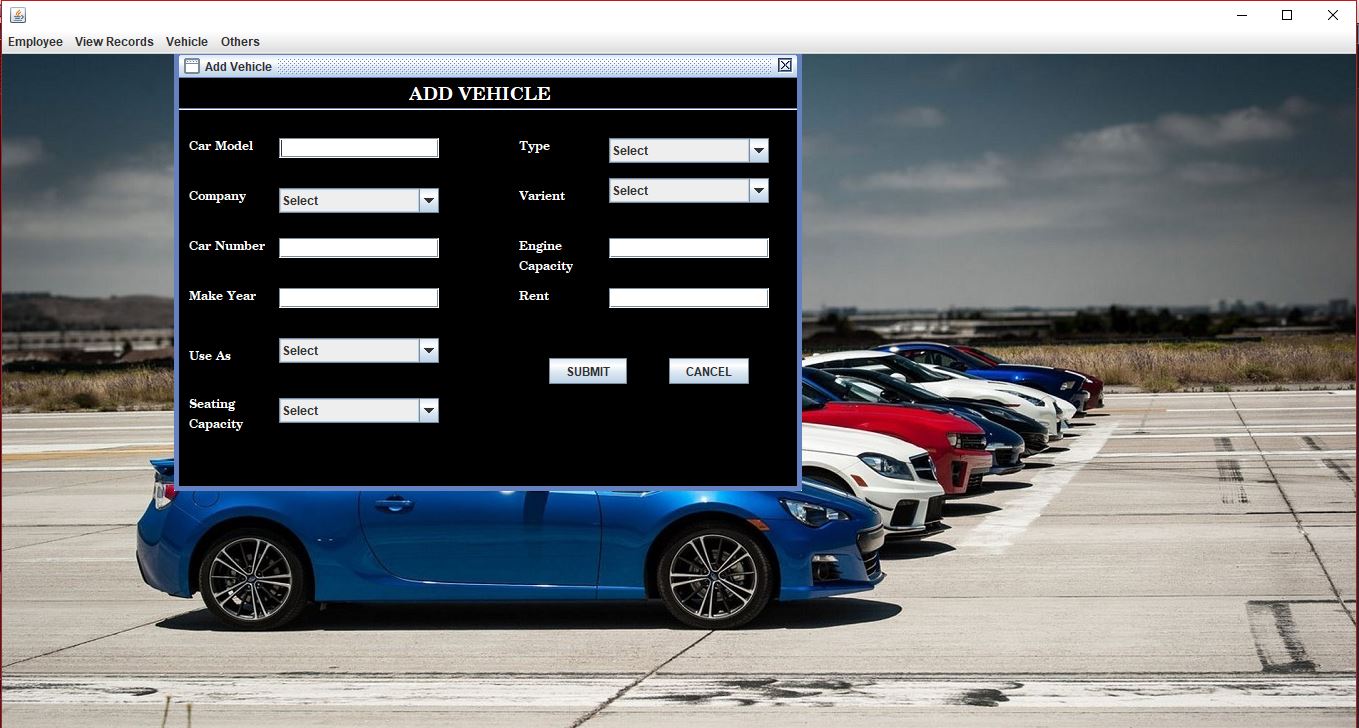
**View Cars: -**



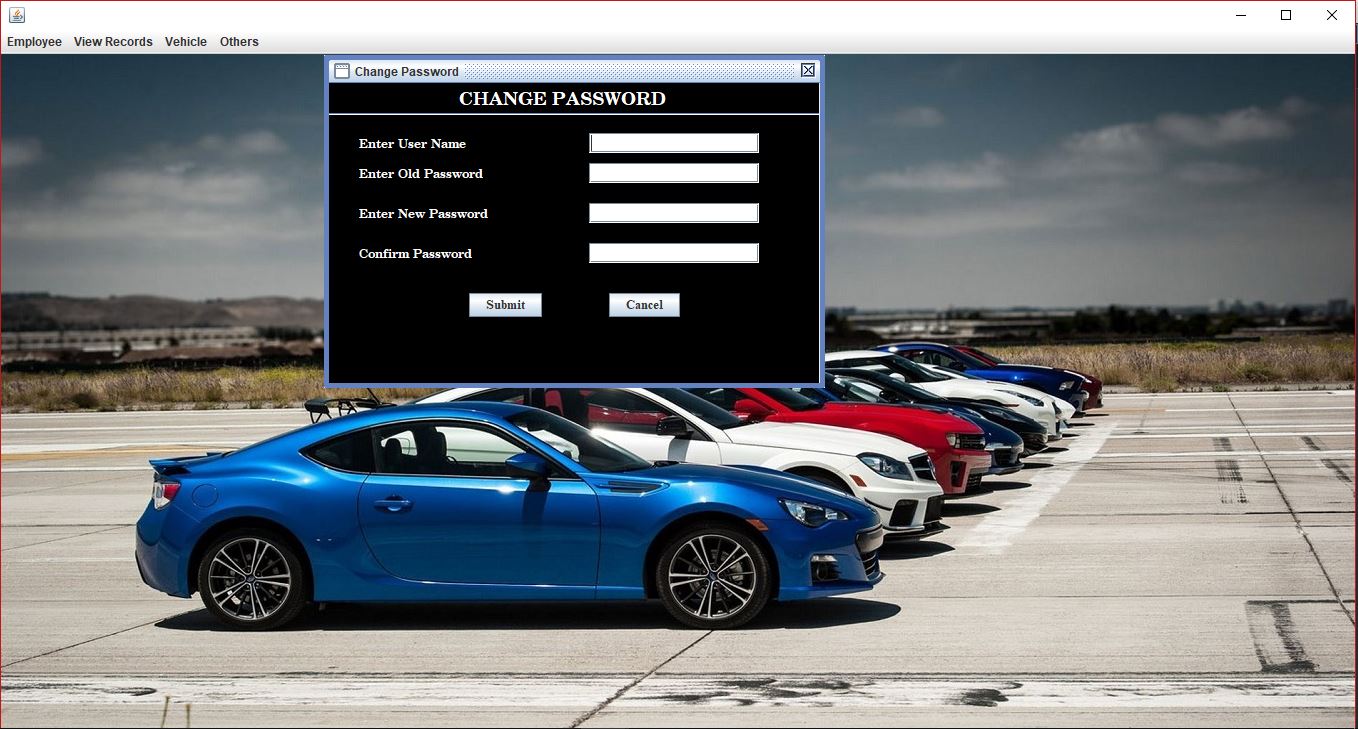
**Add Company: -**



**Add Vehicle: -**



**Change Password: -**



TESTING

**System Testing: -**

Black box testing method was used for system testing. The black box testing usually demonstrates that software functions are operational; that the input is properly accepted and the output is correctly produced; and that integrity of external information (databases) is maintained.

**Why testing is done**

* Testing is the process of running a system with the intention of finding errors.
* Testing enhances the integrity of a system by detecting deviations in design and errors in the system.
* Testing aims at detecting error-prone areas. This helps in the prevention of errors in a system.
* Testing also add value to the product by confirming to the user requirements.

**Causes of Errors**

* The most common causes of errors in a software system are:
* **Communication gap between the developer and the business decisionmaker:** A communication gap between the developer and the business decision maker is normally due to subtle differences between them. The differences can be classified into five broad areas: Thought process, Background and Experience, Interest, Priorities, Language
* .**Time provided to a developer to complete the project:** A common source of errors in projects comes from time constraints in delivering a product. To keep to the schedule, features can be cut. To keep the features, the schedule can be slipped. Failing to adjust the feature set or schedule when problems are discovered can lead to rushed work and flawed systems.
* **Over Commitment by the developer:** High enthusiasm can lead to over commitment by the developer. In these situations, developers are usually unable to adhere to deadlines or quality due to lack of resources or required skills on

the team.

* **Insufficient testing and quality control:** Insufficient testing is also a major source of breakdown of e-commerce systems during operations, as testing must be done during all phases of development.
* **Inadequate requirements gathering:**  A short time to market results in developers starting work on the Web site development without truly understanding the business and technical requirements. Also, developers may create client-side scripts using language that may not work on some client browsers.
* **Keeping pace with the fast changing Technology:** New technologies are constantly introduced. There may not be adequate time to develop expertise in the new technologies. This is a problem for two reasons. First, the technology may not be properly implemented. Second, the technology may not integrate well with the existing environment.

**Testing Principles**

* To discover as yet undiscovered errors.
* All tests should be traceable to customer’s requirement.
* Tests should be planned long before the testing actually begins.
* Testing should begin “in the small” & progress towards “testing in the large”.
* Exhaustive Testing is not possible.
* To be most effective training should be conducted by an Independent Third Party

**Testing Objectives**

* Testing is a process of executing a program with the intent of finding errors.
* A good test case is one that has a high probability of finding an as yet undiscovered error.
* A successful test is one that uncovers an as yet undiscovered error.

**Kinds of Testing:**

**Black Box Testing- Not** based on any knowledge of internal designs or code. Tests are based on requirements and functionality.

**White Box Testing-** Based on the knowledge of the internal logic of an application’s code. Tests are based on coverage of code statements, branches, paths and statements.

**Unit Testing-** The most ‘micro’ scale of testing; to test particular functions and code modules. Typically done by the programmer and not by the testers, as it requires detailed knowledge of the internal program design and code. Not always easily done unless the application has a well-designed architecture with tight code; may require developing test driver modules or test harnesses.

**Integration Testing-** Testing of combined parts of an application to determine if they function together correctly. The ‘parts’ can be code modules, individual applications, client and server applications on a network, etc. This type of testing is especially relevant to client/ server and distributed systems.

**Functional Testing-** Black-box type testing geared to functional requirements of an application; testers should do this type of testing. This doesn’t mean that the programmers shouldn’t check that their code works before releasing it.

**Regression Testing-** Re-testing after fixes or modifications of the software or its environment. It is difficult to determine how much re testing is needed, especially near the end of the development cycle. Automated testing tools can be especially useful for this type of testing.

**Acceptance Testing-** Final testing based on the specifications of the end user or customer or based on use by end-users/ customers over some limited period of time.

**User Acceptance Testing-** Determining if software is satisfactory to an end user customer

**Testing Technique Used**

We will continuously test our project to ensure that it is fully functional. In order to perform testing test cases are designed with the intent of finding the errors in the project and help in removing those errors. Testing begins at the module level and is conducted systematically. It is generally conducted by independent test groups or third party.

Testing is done in our project Vehicle Renting System with help of black box testing that exercise all the functional requirement of the project test cases are designed using this approach by providing set of input conditions to get the expected output.

FUTURE & SCOPE

**1.3.1 Purpose: -**

The purpose of this document is to specify requirements and to give guidelines for the development of above said project. In particular it gives guidelines on how to prepare the above said project.

The advancement in Information Technology and internet penetration has greatly enhanced various business processes and communication between companies (services provider) and their customers of which car rental industry is not left out.

This Vehicle Rental System is developed to provide the following services:

• **Enhance Business Processes:** To be able to use internet technology to project the rental company to the global world instead of limiting their services to their local domain alone, thus increase their return on investment (ROI).

• **Online Vehicle Reservation:** Tools through which customers can reserve available vehicles online prior to their expected pick-up date or time.

• **Customer’s registration:** A registration portal to hold customer’s details, monitor their transaction and used same to offer better and improve services to them.

• **Group bookings:** Allows the customer to book space for a group in the case of weddings or corporate meetings (Event management).

**1.3.2 Scope: -**

This project traverses a lot of areas ranging from business concept to computing field and required to perform several researches to be able to achieve the project objectives.

The area covers include:

•**Car rental industry:** This includes study on how the car rental business is being done, process involved and opportunity that exist for improvement.

• Java Technology used for the development of the application.

• General customers as well as the company’s staff will be able to use the system effectively.

• Web-platform means that the system will be available for access 24/7 except when there is a temporary server issue which is expected to be minimal

• Eco-friendly: The monitoring of the vehicle activity and the overall business becomes easy and includes the least of paper work.

• The software acts as an office that is open 24/7. It increases the efficiency of the management at offering quality services to the customers.

• It provides custom features development and support with the software.

Bibliography

**References**

The following books were referred during the analysis and execution phase of the project

**Common Language Runtime**

By Steven Pratschner

**SOFTWARE ENGINEERING**

By Roger S. Pressman

**UNIFIED MODELING LANGUAGE**

By Gradi Booch, Ivar Jacobson,

James Rambaugh

**COMPLETE REFERENCE .NET**

By David S Platt

**MSDN 2003**

By Microsoft

**IMAGES**

Google Search

Msn Search

Bing Search

**HTML PUBLISHING BIBL**

- Alan Simpson.

**C# 2008**

Andrew Troelson

**WEBSITES:**

[www.google.com](http://www.google.com)

[www.bing.com](http://www.bing.com)

[www.msn.com](http://www.msn.com)

[www.wikipedia.com](http://www.wikipedia.com)

[www.imagegallery.com](http://www.imagegallery.com)

[www.sql.com](http://www.sql.com)’

[www.learnjava.com](http://www.learnjava.com)